DECENTRALIZED APPLICATIONS USING BLOCKCHAIN MANAGING GOVERNMENT CORRUPTION

ABSTRACT

A Blockchain is a continuously growing record, called blocks, which are linked and secured using cryptography such as hashing. Each block contains a hash pointer as a link to the previous block, a timestamp and transaction data. By design, blockchains are inherently resistant to modification of the data. It is similar to a ledger, but in this case, it is a decentralized ledger which can record transactions between two different parties efficiently in a verifiable and permanent way. The data stored in the block is permanently stored and it cannot be modified or deleted. A peer to peer system is used for implementing the distributed ledger. Since blockchains are secure and tamper proof, decentralized consensus has been achieved using blockchains. This technology is used widely in implementing digital currency, decentralized currency such as bitcoin. Bitcoin is a crypto currency and a world-wide payment system. It is the first decentralized digital currency. The network is peer to peer and thus the transaction takes place directly between the user without a centralized body to govern the transaction. These transactions are verified by nodes in the network by the use of cryptography and recorded in the blockchain as the proof of work. The underlying technology used in Bitcoin is blockchains. Blockchain are secure by design and have high Byzantine fault tolerance (resistance of distributed computing systems towards electronic components failure). This makes blockchain potentially suitable for the recording of events, medical record, and other record management activities such as identity management, transaction processing, document processing etc. A blockchain is managed autonomously using a peer to peer network and a distributed time stamping server. Blockchain solves the long-standing problem – double spending. Originally blockchain was developed for accounting bitcoin. Now a days blockchains are used in variety of commercial applications. In this project using the blockchain technology a decentralized application or a database is implemented to record the fund allocation to states by central government so that corruption can be stopped.

LIST OF CONTENT

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	i
	LIST OF TABLES	v
	LIST OF FIGURES	v
	LIST OF ABBREVIATIONS	vi
1	**INTRODUCTION**	
	1.1 Overview of the project	1
	1.2 Need for the project	2
	1.3 Objective of the project	5
	1.4 Scope of the project	5
2	**LITERATURE SURVEY**	
	2.1 Blockchain Technology as an Enabler of Service Systems: A Structured Literature Review	7
	2.2 A review of the Blockchain Literature	10
	2.3 Blockchain and its Application – A detailed survey	14
3	**FEASIBILITY STUDY**	31
4	**4.1 EXISTING SYSTEM**	32
	4.2 PROPOSED SYSTEM	33
5	**SYSTEM DESIGN**	
	5.1 System architecture design	36
	5.2 Data Flow Diagram	36
6	**UML DIAGRAMS**	
	6.1 Use case Diagram	49

6.2 Class Diagram 41

6.3 Sequence Diagram 41

6.4 State chart Diagram 42

6.5 Activity Diagram 43

6.6 Component Diagram 43

7 REQUIREMENT SPECIFICATION

7.1 Hardware requirements 44

7.2 Software requirements 44

7.3 Features of Eclipse IDE 44

7.4 Features of Google - gson 44

8 IMPLEMENTATION

8.1 MainBlock 46

8.2 Block 51

8.3 File hasher 53

8.4 Blockchain Integrity 55

8.5 Get file hash 57

8.6 SHA-256 58

9 TESTING AND MAINTENANCE

9.1 Levels of Testing 62

9.2 Unit testing Implementation 64

9.3 Maintenance 72

10 CONCLUSION AND FUTURE ENHANCEMENTS 75

11 REFERENCES 77

12 APPENDIX 79

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
9.2.1	Test cases	72

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
2.2.1	Overview of blockchain	11
5.1.1	System Architecture Diagram	36
5.2.1	Data Flow Diagram level 0	37
5.2.2	Data Flow Diagram level 1	37
6.1.1	Use case diagram	40
6.2.1	Class diagram	41
6.3.1	Sequence diagram – creation of blockchain	41
6.3.2	Sequence diagram – blockchain integrity	42
6.4.1	State chart diagram	42
6.5.1	Activity Diagram	43
6.6.1	Component Diagram	43
8.1.1	Main block – block creation	50
8.1.2	Main block – block creation	50
8.2.1	Blockchain Block	52
8.2.2	Blockchain	52
8.3.1	File hash generation	55
8.4.1	Blockchain integrity – Valid Blockchain	57
8.4.2	Blockchain integrity – Invalid Blockchain	57
9.2.1.1	Test case 1	68
9.2.1.2	Test case 2	68

9.2.2.1 Test case 3 71
9.2.2.2 Test case 4 71

LIST OF ABBREVIATION

Symbol	Expansion
SHA	Secure Hashing Algorithm
ASIC	Application Specific Integrated Circuit
UTXO	Unspent TransaXtion Output
BTC	BiTCoin
PoW	Proof of Work
P2P	Peer 2 Peer

INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 Overview of the Project

We are living in a digital era where everything is digitized. We are dependent on technology to get our work done fast and efficiently. Most of the systems like file handling, management of transactions, business management etc., are all managed efficiently using technology. In the past file handling and management of transaction (transaction of money, business deals, corporation data etc.,) where done manually and there was a problem of duplicate data which may lead corruption of data. There was also double spending problem in banking system due to improper management of transaction data. All these transactions and management of data was done by a central server or a central system. Thus this may also lead to a risk in which loss of data occurs when the central server crashes. We overcame all these problems slowly using technology and digitized all these process. But still there are problems like duplication and corruption of data, illegal modification of data, money laundering etc. Since all the transaction and data handling process is done by a single body there is less trust placed on this system since there are ways in which people can tamper with the data for their needs. Thus this had lead to corruption of people and government as sometimes there is no proof of the data being tampered or corrupted. To avoid all these problems as much possible we can use Blockchain technology to handle storage of sensitive data and to conduct transactions.

Blockchain is a type of distributed ledger for maintaining a permanent and tamper-proof record of transactional data. A blockchain functions as a decentralized database that is managed by computers belonging to a peer-to-peer (P2P) network. Each of the computers in the distributed network maintains a copy of the ledger to prevent a single point of failure (SPOF) and all copies are updated and validated simultaneously.

A blockchain ledger consists of two types of records: individual transactions and blocks. The first block consists of a header and data that pertains to transactions taking place within a set time period. The block's timestamp is used to help create an alphanumeric string called a hash. After the first block has been created, each subsequent block in the ledger uses the previous block's hash to calculate its own hash. Before a new block can be added to the chain, its authenticity must be verified by a computational process called validation or consensus. At this point of the blockchain process, a majority of nodes in the network must agree the new block's hash has been calculated correctly. Consensus ensures that all copies of the distributed ledger share the same state. Once a block has been added, it can be referenced in subsequent blocks, but it cannot be changed. If someone attempts to swap out a block, the hashes for previous and subsequent blocks will also change and disrupt the ledger's shared state. When consensus is no longer possible, other computers in the network are aware that a problem has occurred and no new blocks will be added to the chain until the problem is solved. Thus, in this project blockchain is used to maintain the integrity of sensitive government data and transaction in tamper proof method. Each transaction and taking place between government bodies and confidential files can be stored using blockchain. The blockchain is shared with every government member so that even in one person tries to modify a document, transaction or confidential files, it can be easily found out and thus the corruption can be stopped to an extent.

1.1 Need for the project

In present day there are lots of corruption taking place in government like illegal documentation, money laundering, alteration of sensitive data etc., Government members are altering transaction and fund data for their own needs and make profit out of it. Most of the profit is black money. Even after demonetization process which took place in India on 8 November 2016, there are still corruption and money laundering taking place. To reduce all these problems, we can use blockchain technology to handle the storage of sensitive files, data and transactions taking place between government system. As blockchain system is distributed in nature, everyone who is involved in using it, has all the data stored in the blockchain and thus even if a single person modifies the data illegally, we can verify that the data is altered. Thus, this helps in preventing future illegal alteration of sensitive data and thus corruption can be reduced.

Corruption is an issue that adversely affects India's economy of central, state and local government agencies. Not only has it held the economy back from reaching new heights, but rampant corruption has stunted the country's development. A study conducted by Transparency International in 2005 recorded that more than 92% of Indians had at some point or another paid a bribe to a public official to get a job done. Earlier, bribes were paid for getting wrong things done, but now bribe is paid for getting right things done at right time. Further, corruption has become something respectable in India, because respectable people are involved in it. Social corruption like less weighing of products, adulteration in edible items, and bribery of various kind have incessantly prevailed in the society. In today's scenario, if a person wants a government job he has to pay lakhs of rupees to the higher officials irrespective of satisfying all the eligibility criteria. In every office one has either to give money to the employee concerned or arrange for some sources to get work done. There is adulteration and duplicate weighing of products in food and civil supplies department by unscrupulous workers who cheat the consumers by playing with the health and lives of the people. In the assessment of property tax, the officers charge money even if the house is built properly according to the Government rules and regulations.

Political corruption is worst in India. The major cause of concern is that corruption is weakening the political body and damaging the supreme importance of the law governing the society. Nowadays politics is only for criminals and criminals are meant to be in politics. Elections in many parts of the country have become associated with a host of criminal activities. Threatening voters to vote for a particular candidate or physically prevent voters from going in to the polling booth – especially weaker sections of the society like tribals, dalits and rural woman occurs frequently in several parts of the country. Recently, the Government increased the salary of the M.P.'s from Rs.16, 000 to Rs.50, 000, that is 300% increase to the existing salary. But many of them are unhappy with rise and want the Government to increase the salary to a much more extent. This clearly shows how the politicians are in constant thirst for monetary benefits and not caring about the welfare of the people. Tax evasion is one of the most popular forms of corruption. It is mostly practiced by Government officials and politicians who lead to the accumulation of black money which in turn spoils the moral of the people.

All these corruptions occur mainly due to alteration of government records illegally to earn money out of it. We can use blockchain technology to store and access government records so that the records are tamper proof and if they are modified we can easily find that the record has been modified and the necessary actions can be taken. This project mainly focuses on the storage of sensitive data which need to be tamper proof. Since the data are stored in blockchain and are distributed to all the users who are involved in using the data, one can easily monitor what transactions are going between the users and see to that the data is not altered illegally. There is transparency in every transaction taking place. The integrity of each transaction is checked by all the users whenever a new transaction occurs and is stored in the blockchain. In this case, illegal alteration of data is reduced. Thus, when this technique is used in transactions involved in government data, the transparency of the transactions increases and thus illegal alteration of data can be avoided. When transactions and storage of data is transparent, corruption can be reduced to an extent.

1.2 Objective of the project

The objective of this project is to improve the transparency of transactions and storage of data using blockchain technology, prevent modification or alteration of transactions and stored sensitive data illegally, to create a tamper proof storage of transactions and sensitive data which can be stored in a distributed manner which improves transparency and avoid single point of failure of the system and reduce corruption in government by implementing the above said features by conducting transactions and storage of government data using blockchain technology in a decentralized way so that not a single body is responsible for managing all the transaction and storage process, instead everyone is involved in the said process.

1.3 Scope of the Project

The scope of this project is to develop a way to conduct and store government transactions in a more transparent way using blockchain technology. Transactions are conducted and stored in a decentralized manner thus everyone using this application are involved in maintaining the integrity of the transaction and the data. The project also involves in storing tamper proof data and avoids single point of failure in the system.

LITERATURE SURVEY

CHAPTER 2
LITERATURE SURVEY

2.1 Blockchain Technology as an Enabler of Service Systems: A Structured Literature Review

Author: Stefan Seebacher(&) and Ronny Schüritz

Karlsruhe Institute of Technology, Kaiserstr. 89, 76131 Karlsruhe, Germany

Abstract

Blockchain technology is expected to revolutionize the way transactions are performed, thereby affecting a vast variety of potential areas of application. While expectations are high, real world impact and benefit are still unclear. To be able to assess its impact, the first structured literature review of peer-reviewed articles is conducted. As blockchain technology is centered around a peer-to-peer network, enabling collaboration between different parties, the service system is chosen as unit analysis to examine its potential contribution. We have identified a set of characteristics that enable trust and decentralization, facilitating the formation and coordination of a service system. Blockchain technology is known as the underlying basis of Bitcoin. Apart from its utilization in the Bitcoin network, many researchers and practitioners expect it to generally revolutionize the way we interact and transact over the Internet, resulting in the dawn of a new economy. A vast potential for its application is predicted, for example affecting the way governments, public notary services or contracts in an online environment work. Expectations towards the potential of this new technology are rising, which can be seen in Gartner's Hype Cycle, where blockchain technology has already reached the peak of inflated expectations. But as the term inflated expectations indicates, there is a difference between expectations and experienced real-world impact. In that context, Gideon Greenspan, the CEO of

Multichain a blockchain provider, is stating that businesses are still “waiting to gain a clearer understanding of where blockchains genuinely add value in enterprise IT”. While, there are several start-ups, that already offer blockchain solutions to their customers, no application has yet achieved large scale recognition, as they face competition of existing and well-established systems. Therefore, additional and pervasive use cases are needed to foster the adoption of blockchain technology and to reveal real world benefits for its users.

In order to facilitate the identification of practical use cases, it is necessary to be aware of potential impacts, which result from the application of blockchain technology. As it is built upon interaction in networks or systems, we investigate its implications in the context of service systems, which themselves are characterized by collaborative processes and, therefore, serve as an excellent unit of analysis.

Performing the first structured literature review on blockchain technology, which is entirely based on peer-reviewed literature, we derive a distinct set of characteristics that we illustrate in a concept matrix and interpret. The characteristics are then assessed concerning their contribution to service systems, developing a better understanding of the potential of blockchain technology.

Review on the Blockchain technology

In this section, we present the results of our literature review on blockchain technology. We start by formulating a definition for the basic concept, which is followed by a presentation of the technology’s inherent characteristics.

The concept of Blockchain Technology

Although blockchain technology was first introduced in the year 2008 in Nakamoto’s white-paper as the underlying technology of Bitcoin, a generally accepted definition of the concept has not been established. Therefore, this section, provides a definition of the concept based on peer-reviewed literature.

While some authors refer to a blockchain as a distributed data structure, database or system, others call it a decentralized network. Serving as a log or ledger to document all transactions and activities that took place within the construct, it is a linked sequence of transactions, in which time-stamped transactions are broadcasted to and shared with participating entities, located in its belonging peer-to-peer network. Transactions are secured through public-key cryptography and verified by the participants for correctness. Once a transaction is verified by the participatory community, it is added to an unpublished block. Amongst others, a block serves as storage unit for transactions and contains a reference to the settled and verified chain of blocks. Through the use of a consensus mechanism new blocks are added to the blockchain in an append-only manner and then cannot be altered anymore. Based on the presented statements, we synthesize the following definition for a blockchain:

A blockchain is a distributed database, which is shared among and agreed upon a peer-to-peer network. It consists of a linked sequence of blocks, holding timestamped transactions that are secured by public-key cryptography and verified by the network community. Once an element is appended to the blockchain, it can not be altered, turning a blockchain into an immutable record of past activity.

References

1. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)

2. Swan, M.: Blockchain: Blueprint for a New Economy. O'Reilly Media, Inc., Sebastopol (2015)

3. Tapscott, D., Tapscott, A.: Blockchain Revolution. Penguine Random House, New York (2016)

4. Olnes, S.: Beyond bitcoin enabling smart government using blockchain technology. In: Scholl, H.J., et al. (eds.) EGOVIS 2016. LNCS, vol. 9820, pp. 253–264. Springer, Cham (2016).

2.2 A review of the Blockchain Literature

Author: George Pîrlea, george.pirlea.15@ucl.ac.uk November 14, 2016

Bitcoin

On October 31st, 2008, Satoshi Nakamoto announces that he has published a design paper for Bitcoin, "a new electronic cash system that's fully peer-to-peer, with no trusted third party". Block 0 of the Bitcoin blockchain, the Genesis Block, is established at 18:15:05 GMT on January 3rd, 2009. The block contains the message "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks". Version 0.1 of the Bitcoin software was released on January 9th, 2009. The current system of electronic payments relies on trusted financial institutions to process transactions. This *trust-based* model has inherent weaknesses:

- Non-reversible transactions are not possible. disputes need to be mediated
- Small payments are infeasible; cost of mediation increases transaction costs
- Transactions are not anonymous; third party must collect personal information

Bitcoin, on the other hand, is an electronic payment system based on cryptographic proof of work instead of trust, allowing two parties to transact directly with each other without the need for a trusted third party.

Technical Analysis

The Bitcoin paper gives a very high-level overview of the system. It does not describe how Bitcoin is actually implemented. If you only read the paper, you will have an inexact (and in some cases incorrect) view of how Bitcoin works. For example, Bitcoin uses a stack-based scripting language for transactions, but the paper does not mention it at all. To get an accurate understanding of Bitcoin internals, check the source code, the developer guide and *Mastering Bitcoin* by Andreas Antonopoulos.

Blockchain

Bitcoin keeps a public ledger of all transactions, called the *block chain*. This is, in essence, a peer-to-peer distributed timestamp server. Its role is to prevent double spending and modification of previous transaction records. Each full node in the Bitcoin network keeps a complete copy of the block chain, containing all blocks validated by that particular node. Consistency across the network is guaranteed by following a series of *consensus rules*. When several nodes in the Bitcoin network independently arrive at identical block chains, they are considered to be in *consensus*.

Overview

A block chain, as the name suggests, is a digital chain of *blocks*. Each block contains a timestamp, a *nonce* (a number whose purpose will be described later)

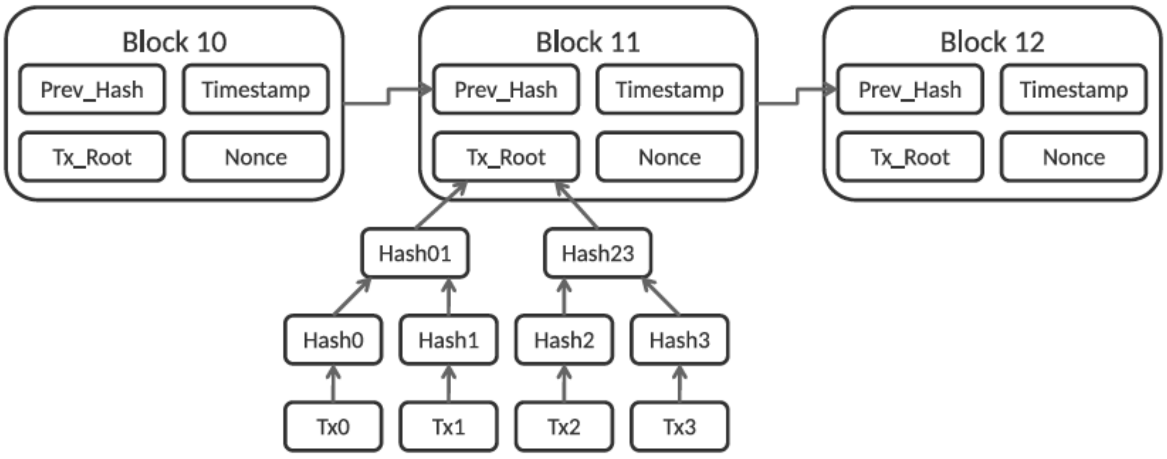


Fig 2.2.1

and a Merkle Tree that stores transactions. The blocks are cryptographically chained together using hashes. Each block contains the hash of the previous block, which in turn contains the hash of the previous, and so on back to the first block. Modifying any part of a past block would invalidate all the subsequent hashes. This is critical for preserving the integrity of the ledger. Computing a hash, however, is not very difficult. A usual CPU can compute roughly a million

SHA-256 hashes per second. GPUs can compute hundreds of millions of hashes per second. ASICs (application-specific integrated circuits) are even faster. Under normal circumstances, it would be possible to modify previous blocks, as you could simply recompute all subsequent hashes.

Proof of work

The block chain is maintained collaboratively by peers on the Bitcoin network. To prevent malicious peers from modifying past blocks, the network requires a significant amount of work to be invested in the generation of each block. Chaining blocks together makes it impossible to modify transactions included in any block without modifying all following blocks. When there are multiple chains competing for acceptance in the network (for example, when someone tries to rewrite a past block), consensus rules dictate that nodes will pick as valid the chain that is longest, i.e. the chain that has most proof of work behind it. To successfully rewrite a past block, a malicious actor needs to re-generate all following blocks and overtake the honest nodes in the network. This is computationally infeasible, as long as honest peers control a large share of the total hashing power.

Implementation

The network only accepts a block as valid if the hash of its header is less than a certain *hash target*, which is computed based on the network *difficulty*. The network sets the difficulty and updates it every 2016 blocks so a block is generated, on average, every 10 minutes. Remember the nonce in the block header that we did not explain? This is where it comes in. The nonce is a 32-bit integer that you can increment to change the block's hash. When mining, you try different nonces until the block header hashes to something less than the hash target. In practice, since the network difficulty is high, a 32-bit nonce is not enough to beat the hash target, so miners also change extra Nonce, a field included in the Coinbase transaction of the block. As the Coinbase transaction is included in the Merkle Tree, any change propagates upwards into the Merkle root in the header. Bitcoin uses the Hash cash proof-of-work function.

Ethereum

The Bitcoin protocol does facilitate a weak version of *smart contracts*, however the scripting language implemented in Bitcoin has several important limitations:

- **Lack of Turing-completeness** – the scripting language doesn't have loops
- **Value blindness** – cannot control what amount is withdrawn; either the entire UTXO is withdrawn, or none of it is
- **Lack of state** – UTXOs scripts can only be used for one-off contracts; "state-full" multi-stage contracts such as decentralized organizations are impossible to implement
- **Blockchain blindness** – cannot access blockchain data such as block headers or transaction history.

Rationale

Ethereum aims to build "the ultimate abstract foundational layer": a blockchain with a built-in Turing-complete programming language, value awareness, blockchain awareness and state. Smart contracts, decentralized applications and even entirely new protocols (currencies, reputation systems etc.) can be developed on top of the Ethereum block chain. This new wave of blockchain-based technology is called "Blockchain 2.0".

Technical Analysis

For a high-level overview of Ethereum, the original white paper by Vitalik Buterin is a good resource. The Ethereum Yellow Paper by Gavin Wood provides a formal specification of the system. Indeed, this was the first formal specification of *any* blockchain protocol.

References

1. Satoshi Nakamoto. Bitcoin P2P e-cash paper. Oct. 2008. url: http://www.metz dowd.com/pipermail/cryptography/2008-October/014810.html.
2. Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. url: https://bitcoin.org/bitcoin.pdf.
3. Block #0. url: https://blockexplorer.com/block/000000000019d6689c085a e165831e934ff763ae46a2a6c172b3f1b60a8ce26f.
4. Genesis block. Bitcoin Wiki. url: https://en.bitcoin.it/wiki/Genesis_bloc k.
5.Satoshi Nakamoto. Bitcoin v0.1 released. Jan. 2009. url: http://www.metzdowd.com/pipermail/cryptography/2009-January/014994.html.

2.3 Blockchain and its Application - A detailed survey

Author: Supriya Thakur Aras, Department of Computer Engineering, Maharashtra Institute of Technology, Pune, India
Vrushali Kulkarni, PhD, Department of Computer Engineering, Maharashtra Institute of Technology, Pune, India

Abstract

Blockchain is being termed as the fifth disruptive innovation in computing. In simplest words, it is a distributed ledger of records that is immutable and verifiable. Since its advent in 2008, blockchain as a concept has been used in various ways. The largest impact or application is seen as a multitude of cryptocurrencies that have sprung up. However, with time, it has become clear that blockchain as a technology is likely to have an impact much wider than just the cryptocurrency domain and much deeper than simple distributed ledger storage. This detailed survey intends to bring together all the key developments

so far in terms of putting blockchain to practice. While the most common adoption of blockchain is in finance and banking domain, there are experiments being conducted by many big players in various other domains. This paper will explore the various domains where blockchain has had an impact and where future implementations may be expected.

Introduction

Blockchain technology or the distributed, secure ledger technology has gained much attention in recent years. This paper presents a detailed survey of blockchain technology literature and its applications. The sources of blockchain literature examined for this survey include research papers, books and book chapters, journal papers, specific cryptocurrency sites and wikis, conference papers, company 'Point of View's (PoVs), white-papers published by various organizations implementing and experimenting in Blockchain. Blockchain being a much hyped and experimented technology a lot of literature is found in content hosted on proprietary forums such as company websites, web articles, etc. This survey is extensive and covers the various aspects of blockchain including consensus algorithms and their variations as well as currently implemented and possible future applications. This survey will not cover the details of technical aspects of blockchain, however, references that cover these aspects may be found in bibliography.

Blockchain Overview - Technology

A very significant plus of the blockchain technology is that it solves two of the most dreaded problems of currency-based transactions, which have so long necessitated the requirement of a third party to validate the transactions. These are popularly known as the Byzantine Generals' Problem and the Double Spend Problem. With advent of Blockchain, crypto-economics has evolved.

This very aspect has been highlighted in works of Pilkington. The paper explains how blockchain as a concept can be applied to a non-tokenized scheme. The paper also talks about blockchain taxonomy and how hybrid solutions become an obvious choice and moving from permission-less to permission blockchain becomes imperative to solve certain kinds of problems where trust is paramount and a public permission-less ledger seems both a risk and an overhead.

For a long time, the terms Bitcoin (cryptocurrency which has been the first and the most successful of the blockchain based cryptocurrencies) and Blockchain have been used interchangeably. Swan explains how these terms were used to mean one of the three things – firstly the underlying blockchain technology platform, secondly the Bitcoin protocol i.e. the software which actually runs on the Bitcoin blockchain's network computers and makes transactions possible and thirdly the Bitcoin digital currency itself (denoted as BTC) which is the source of value. The three things listed above can be visualized as the layers of the Blockchain stack. Blockchain technology forms the lowermost layer with the Bitcoin protocol in the middle and the digital currency forming the top layer. Swan's book considers the evolution of Blockchain technology in three generations. The digital currency application is considered the Blockchain 1.0, the application of blockchain to smart contracts and Distributed Applications (DApps) is considered the Blockchain 2.0 and finally the application beyond currency and economics is detailed as the Blockchain 3.0.

Peters and Panayi provide a comprehensive overview of emerging blockchain architectures, their distinction from traditional databases and role of blockchain in electronic exchanges.

Blockchain Protocols

Blockchain eliminates the need for third party to conduct transactions on one's behalf. This implies that the consensus mechanism has to exist in the network itself. How a given blockchain network implements its consensus mechanism, determines the strength of the network. A foolproof consensus mechanism, suitable for purpose (of the blockchain in question) is essential to maintain sanity and coherence of data among the participating nodes of the network. The consensus mechanisms of blockchain aim to eliminate mainly two known problems with digital currency - Remove the problem of double spend and Eliminate Byzantine Generals problem.

While much work has been done on blockchain protocols, there are some key algorithms explained in brief here whose variations are being used and further developed to suit various applications of blockchain. Cachin et al. have explained blockchain consensus mechanism and various consensus algorithms in their research paper.

Proof of work

PoW protocol requires all nodes on the network to solve cryptographic puzzles by brute force. For example, in case of Bitcoin blockchain, the new transactions are tentatively committed and then based on the PoW output, a selected block created by the winning node is broadcast to all the nodes, at specific synchronization intervals. Once the block is transmitted using peer to peer communication to all other nodes, the same is included in the blockchain and any tentative transactions are rolled back. By rule of probability, the consensus is achieved as 51% of power rather than 51% of people count. Effectively the computing power used by all other nodes except the winning node, is wasted.

Proof of stake

Proof of stake protocol of block verification does not rely on excessive computations. It has been implemented for Ethereum and certain altcoins. Instead of splitting blocks across proportionally to the relative hash rates of miners (i.e. their mining power), proof-of-stake protocols split stake blocks proportionally to the current wealth of miners. The idea behind Proof of Stake is that it may be more difficult for miners to acquire sufficiently large amount of digital currency than to acquire sufficiently powerful computing equipment. It is also an energy saving alternative.

A variation of POS is the Delegated Proof of Stake (DPOS) algorithm. Delegated proof of stake (DPOS) is similar to POS, as miners get their priority to generate the blocks according to their stake. The major difference between POS and DPOS is that POS is a direct democratic while DPOS is representative democratic. Stakeholders elect their delegates to generate and validate a block. With significantly fewer nodes to validate the block, the block could be confirmed quickly, making the transactions confirmed quickly. Meanwhile, the parameters of the network such as block size and block intervals could be tuned by the delegates. DPOS is implemented by Bitshares.

Practical Byzantine Fault Tolerance

An approach to deal with the Byzantine Generals problem is the Federated Byzantine Agreement (FBA). In this approach, it is assumed that the participants of the network know each other and can distinguish which ones are important and which ones are not. PBFT (Practical byzantine fault tolerance) is a replication algorithm which utilizes this principle. Hyper-ledger utilizes the PBFT as its consensus algorithm. There are designated validator (primary) nodes that are each associated with a group of nodes. The primary is responsible for multicasting requests to other replicas in its group. A service operation would be valid if it has received approvals from over 1/3 different replicas. Additionally, if a client does not receive the replies, it will send the request to all replicas instead of only sending it to the primary in case the primary is faulty. A primary is responsible for ordering the transaction and each replica commits the transaction in the same order. It has been seen that PBFT or its variations map well to the needs of various organizations like banks, supply chain or payroll systems.

Blockchain Applications

Bitcoin has been the mainstay to many of the other applications of blockchain. Many projects have been implemented to overlay the Bitcoin blockchain as noted by Swan and Crosby et al. This not only makes the Bitcoin more powerful and popular, but also reinforces the notion that Bitcoin is here to stay. Some examples are MasterCoin, NXT, Open Assets, ColoredCoins, etc.

Pilkington in also explains how the concept of blockchain can be extended beyond digital currency to any asset that has a definite value associated with it. The paper explains some of the popular cryptocurrency applications like Ethereum, Ripple, Gridcoin, etc., and also lists possible future applications in various domains such as digital identity provisioning, voting, commodity trading, etc. Interesting insights on Blockchain impact to Financial Domain can be obtained from the Edgeverve Infosys Finacle Report, Feb 2017 Blockchain Technology from Hype to Reality. As per this report derived from a survey of over 75 financial institutions, nearly 50% of the banks surveyed have already invested in Blockchain technology or were likely to do so in 2017. This survey proved that blockchain is being tried in almost all-important domains such as healthcare, finance, supply chain management, reputation management, etc.

Social Inclusion

As internet has become an accessible global platform to bring the world together, thanks to the mobility revolution, it is possible for the people in remotest parts of the world to access internet resources across the world. Cryptocurrencies enable people with no access to physical banks to perform global transactions with others across the world. As sited by Pilkington, thanks to Bitcoin, sellers like Indian handicraft work artisans have now found a global marketplace to sell their work. This takes away the hassle of fiat foreign currency availability and conversion. Bitcoins are an easily accessible, usable global cryptocurrency which provides value for their work.

Cryptocurrency

Currency that is in use across the world is largely fiat currency or currency whose value is assured by a government guarantee, e.g. Indian Rupees, US Dollar, Great Britain Pound, etc. These currencies are not backed by physical assets. Commodity money is backed by a tradable resource, like Gold and Silver. Its value is at least as much as the value of the commodity itself.

Cryptocurrency such as Bitcoin does not fit into any of the above categories. Cryptocurrencies are a medium of exchange that uses cryptography to secure transactions. They are a poor store of value compared to traditional fiat currencies and have lower price stability due to lack of government intervention. However, cryptocurrencies are a more efficient medium of exchange as blockchain technology is uniquely positioned to tackle speed and cost. At the time of writing this paper in Dec 2017, over 1300 cryptocurrencies existed, with a total market capitalization of $ 431,029,932,585. Bitcoin is the most successful and most widely circulated cryptocurrency with a market cap of nearly $24,747,300,000. There are many cryptocurrencies being created and used for specific purposes. It may be noted that the value of the cryptocurrency is measured using the fiat currency.

Private data storage

A generic extension of blockchain transactions to transfer stuff other than cryptocurrency is suggested by Zyskind et al. In their proposed system, the transactions are used to carry instructions for storing, queuing and sharing data. With increased number of mobile applications seeking complete access to user data such as contacts, messages, photos and a variety of other personal data, Zyskind et al. have provided the implementation architecture of a system which uses blockchain along with an offline storage mechanism in order to manage permissions explicitly for each line item, rather than giving complete access permission indefinitely. Offline storage such as LevelDB or any cloud storage can be used to limit the amount of data stored in the blockchain. This could however result in a limited third party dependency but makes the solution more scalable. Organizations may choose technology upgrade to adopt a more reliable data privacy solution, for their data.

Reputation Management

A successful implementation of reputation management can be found in Accenture's Akshay Patra Midday Meal Program Management project. This project used a private blockchain implementation to gather real time, direct feedback from schools that is not manipulated by intermediaries. Thus blockchain has provided the required transparency to the meal chain, to help in audits and invoicing. This has also saved the manual effort of collecting, collating and transmitting the feedback.

Education

Blockchain can be the transformational force in education as well. Sharples and Domingue have suggested the use of blockchain to provide a verifiable, easily shareable and permanent record of such educational records and rewards. It also talks about the possibility of having an 'Educational Reputation Currency', which is initially distributed to participating institutes based on any existing metric. This currency can then be propagated successively in the blockchain and may be awarded to promote learner reputation.

One limitation not completely addressed in this paper is how the creation of such a reputation currency shall be controlled. For example, in case of Bitcoin blockchain, Bitcoins are created whenever a block is added to the blockchain. The added Bitcoins are awarded to the node that added the block. The quantity of Bitcoins created is also defined by the Bitcoin algorithm. At the time of writing this paper, every added block adds 25 Bitcoins to the winning node's account. Using an external third party ranking of educational institutes may create a bias and participants may question fairness. A successful implementation of blockchain to award educational certificates has been done by Sony and University of Nicosia.

Banking

The impact of blockchain as a technology was first felt by the banking and trading sector. So much so that Bitcoin and its underlying technology, the blockchain, were initially seen as the biggest threat to banking businesses worldwide. However, in past few years it has been seen that banks have deep dived to make this technology work for them in a favorable manner and are experimenting various ways to use blockchain in their business.

Some experts however still do believe that blockchain will lead to the end of several long-standing businesses and professions. Typical banking processes like approval of a loan or derivative is a time-consuming process due to multiple back end steps involving contract negotiations with multiple parties. Blockchain provides the necessary transparency and speed via smart contracts, to this requirement. Multiple banks are already experimenting Blockchain-as-a-Service offering from technology companies such as R3, IBM and Microsoft.
The potential role of blockchain in banking is dealt with in great detail in. Panayi et al. discuss automation of various niche aspects of banking like client account reconciliations, data loss reporting, Over the Contracts (OTC) contracts/products and clearing settlement, cash management by government, etc.

Finance - Payroll and settlement

Public service transactions may be as trivial as buying a train ticket or more complex ones such as marriage registration, property buy and sell, patent management, etc. Typically, public service transactions require a series of actions to validate the authenticity of the transacting party (or parties), verification of the data provided by the transacting party (or parties), conduct the required transaction and finally provision of the required service followed by recording of the end to end transaction. This translates into significant turnaround time for the transacting parties.

A digital blockchain ledger can reduce this turnaround time to minimum as the most important asset ownership validation and verification is performed taking advantage of the intrinsic nature of the blockchain.

Sestoft proposes a distributed system – Autonomous Pension Fund that would be a self-sustaining running autonomous contract-based system to manage life-based pension funds without a central trusted pension fund. Since a large number of activities related to life-based pension such as receiving payments from active customers, making payments to beneficiaries and payments of taxes on pensions are mainly processing of contract regulated payments, Sestoft opines that they can be executed using Self Executing Contracts and a cryptocurrency. Sestoft has proposed use of Ethereum for the algorithm. A prerequisite here is the fact that such an autonomous system will need event insurance relates life-event triggers from other trusted bodies, so that self-executing contracts can act on them. A key challenge noted by Setsoft for the Autonomous Pension Fund system is the long term nature of the engagements with the customer/beneficiaries till their death. It is challenging for people to have faith in an autonomous system to keep its promises and more so to keep faith in the technology and its sustenance for that long a period. The latter challenge about faith in longevity of the technology itself, equally applies to most of the blockchain applications.

Blockchain for public services

Taxation

As ideated in PWC, UK, report, taxation is one area where blockchain can potentially make a big contribution. The report relates the key attributes of blockchain namely provenance, transparency and traceability to the exact needs of a modern taxation system. A huge advantage of cutting on administrative cost can result from the use of blockchain especially in transaction taxes such as VAT, withholding Tax, stamp duties, etc. In a sharing economy, blockchain could be used to achieve compliance and transparency for tax payments, thus shifting the responsibility of collecting tax from tax authorities to participants of the sharing economy.

In countries like India which are moving towards uniform taxation via GST (Goods and Services Tax), blockchain can help in tracking the end-to-end collection and expenditure of taxes by the government. While the tax provenance aspect is very important and so also is the utilization of tax earnings.

The biggest challenge however, in this would be to achieve digitization of currently non-digital sellers who rely on paper records rather than digital ones. Pilots in this area in various countries are likely to be seen in future.

Healthcare

Over the past decade, healthcare is turning increasingly digital with more and more doctors, hospitals, healthcare machineries going digital to store their patient records. Digitization of medical data enables easy retrieval, sharing on need basis for better decision making based on historic cases and is also very crucial for legal purpose record keeping. However, medical data digitization also exposes it to a bigger risk of patient privacy violation.

A blockchain based Healthcare Data Gateway (HDG) is proposed by Yue et al. They propose the use of a private blockchain cloud to guarantee that the medical data cannot be changed by anybody including the patient himself and/or the physicians. Medical data is diverse in kind, i.e. it could be numeric, textual, image data (scans, x-rays, photos, etc.), video data (transcripts, recordings, etc.), etc. To remove the complexity of storing varied data types Yue et al. propose an Indicator Centric Schema (ICS) based data model. In this model, a single table shall be used to organize all data for a given patient and would include simple relevant fields like timestamp, indicator, type, value and category. The ICS also can be extended to include a Purpose Centric Access Control model which would include say requestor, indicator, timestamp, purpose and retention duration. Such a model is extensible for use in other similar applications of blockchains where data of different types needs to be stored. A segregation of frequent and infrequently accessed data into separate blocks of the blockchain may also be done.

Xia et al. have also proposed a blockchain based system MeDShare, for sharing medical data among cloud service providers. MeDShare would provide data access control, provenance and auditing. The proposed system also used smart contracts for data behavior detection from data access patterns and blocks malicious users.

Voting

In the year 2014, a Danish political party was the first to use blockchain technology for voting [26]. Online voting platforms such as ‘Followmyvote’, which enable digitally secure blockchain based voting have also been created.

Insurance

Cognizant Technology Solutions give an end-to-end view of how blockchain can transform insurance in their perspectives. Travel insurance, crop insurance, property and casualty insurance and most importantly health insurance are all set to change with the use of blockchain technology. A multiparty shared network with insurers, hospitals, funeral homes, a department of health and the beneficiary forming the nodes of the blockchain, can be created. This setup will provide the necessary disintermediation and speed required for the insurance and claim process to be streamlined and to eliminate frauds.

Smart Cities

A possible application of Blockchain to smart cities is suggested by Sun et al. in. Authors relate a smart city to a sharing economy where information and communication technologies are utilised to enhance opportunities of sharing of resources. Author proposes using a blockchain based framework for sharing of resources across various services to ensure data immutability, accountability, proper asset utilisation and to reduce transaction costs.

Other related work

Blockchain has its inherent challenges and limitations. Due to peer to peer network operation, it is high on energy usage and hence wastage per unit computation. While all the network nodes compete to add the block in case of a Proof of Work based blockchain system, only one node succeeds in adding its transactions block each time. As a result, while other blocks contributed to transaction validation and verification process, thus reinforcing it, their efforts are effectively being wasted when the given transaction block is not added.

Wang et al. have assessed many such pitfalls as they have evaluated blockchain against a maturity model. They evaluate blockchain for four technology maturity parameters as defined by ACM Computing Classification System (CCS 2014) and against the five stages of the Capability Maturity Model (CMM). Blockchain's merits over traditional distributed databases are often debated. A school of thought is that distributed databases are a cheaper option with less power wastage. Peters and Panayi provide a lucid distinction between the two.

Blockchain challenges

Regulation is the biggest challenge for non-fiat currency. The rate of technical innovation is surpassing the rate at which regulations catch up. The currency evolution has seen a transformation in the order from fiat currency to e-money to virtual currency to cryptocurrency. Cryptocurrency is the first decentralized version of currency. Some regulatory bodies hold the opinion that cryptocurrency does no fulfil the functions of money primarily due to its value volatility.

It is a challenge more from the governance perspective rather than from the cryptocurrency user's perspective. There are already reports of Bitcoin being used for illegal activities, drug rackets, money laundering, etc. Trevor Kiviat highlights the difference between at currency and cryptocurrency and the challenges associated with cryptocurrencies regulation. IRS of USA have framed laws for taxation of Bitcoin holdings while Russia is considering banning Bitcoin due to the usage of this unregulated currency for unethical purposes. China also has banned Bitcoins while Australia has passed a resolution to accept Bitcoin transactions.

The Economist (2015) article - The magic of mining highlights a very important challenge of power consumption associated with mining and provides with some examples of how increasing power is being invested in mining activities to earn Bitcoins.

Bitcoin's increasing adoption has led to concerns about the ability of the underlying blockchain technology to scale. Since Bitcoin is a self-regulating system that works by discovering blocks at approximate intervals, its largest transaction throughput is effectively capped at maximum block size, divided by the interval. In their paper, Wei Xin et al. propose various strategies to improve private blockchain scalability. The have recommended and experimentally shown that optimization of parameters like block construction, block size, time control and transaction security can lead to better performance and lower error rates. In the light of the fact that several international electronic primary financial exchanges have begun to announce they will explore the adoption of blockchain technology in their trade processing and reporting for execution and clearing, Peters and Vishnia examine the current status of regulatory requirements and the challenges faced by market participants in meeting them.

An interesting tradeoff is revealed by the work by Rimba et al. on cost of storage and computation of business processes on a standard cloud environment vs. blockchain environment. As per the results of this experiment costs of a single business process (Incident Management) were higher on Ethereum blockchain than on Amazon SWF. However, the experiment is done for a limited scope of a single business process and the results may not be generalizable, given the day to day advances in blockchain technology towards its optimization. One key limitation of Blockchain technology is the scalability issue due to size of the public or permission less blockchain. Blockchain optimization and scalability is an area of much research. In Gencer et al. propose a service-oriented sharing technique to achieve blockchain scalability and extensibility.

Conclusion

In a plethora of blockchain based applications and experiments, faith on the longevity of blockchain technology, is increasing. Scalability and consensus algorithms are areas of growing research in order to make blockchain more adaptable for businesses of larger scale. Areas like taxation, education, insurance are yet to see a major overhaul via blockchain adoption and these can be the focus areas of future research in blockchain. Acceptance of cryptocurrency by governments and establishment of regulations governing them are very important to ensure ethical use of cryptocurrency. The public blockchains also provide an opportunity of mining interesting patterns of cryptocurrency usage, user behaviours and monetary networks across the globe.

References

1. Pilkington Mark. 2016 Blockchain Technology: Principles and Applications, Research Handbook on Digital Transformations, Social Science Research Network

2. Swan Melanie. 2015. Blockchain: Blueprint for a new Economy, OReilly Publications

3. Peters G.W. Panayi E. 2016. Understanding Modern Banking Ledgers Through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money, Banking Beyond Banks and Money, Springer Sep 2016, pp. 239-278

4. Satoshi Nakamoto. 2008, Bitcoin: A Peer-to-Peer Electronic Cash System, [Online] http://www.bitcoin.org

5. Buterin, Vitalik. 2015, On Public and Private Blockchains. [Online] https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/

FEASIBILITY STUDY

CHAPTER 3

3.1 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the government organization as the project involves in major change of the current system. The adoption of this project should be seamless and should be carried out with ease and in less time. For feasibility analysis, some understanding of the major requirements and the concepts of how the underlying technology works is essential.

3.2 ECONOMIC FEASIBILITY

This study is carried out to check the economic impact that the system will have on the government organization. The amount of cost required to implement this project is less as only the integrity of the data or the transaction is checked. This project can be imposed on the existing system and making it decentralized, reduces the cost required for installing computer servers. Thus, the developed system is well within the budget as the software used for developing this project is open source.

3.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. The proposed project requires less technical resources for execution as the work load is distributed to all the systems in the network. System which requires high system resources may not be accepted by the client i.e., the government organization. Thus, the system is developed to consume less resource while executing.

3.4 OPERATIONAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. This system requires knowledge on blockchain and how it works for the administrator to monitor the system. The user must have basic knowledge of decentralized systems, P2P systems and their working principles. Minimal knowledge is required for operating the system as most of the process i.e., transactions and storage of sensitive data, is handled by the system itself. The user can operate the system with grate ease.

4.1 EXISTING SYSTEM

In the existing system, all the transactions and the data stored in the blocks goes through Proof of work algorithm. A proof of work is a piece of data which is difficult (costly, time-consuming) to produce but easy for others to verify and which satisfies certain requirements. Producing a proof of work can be a random process with low probability so that a lot of trial and error is required *on average* before a valid proof of work is generated. Bitcoin uses the Hashcash proof of work system.

One application of this idea is using Hashcash as a method to preventing email spam, requiring a proof of work on the email's contents (including the to address), on every email. Legitimate emails will be able to do the work to generate the proof easily (not much work is required for a single email), but mass spam emailers will have difficulty generating the required proofs (which would require huge computational resources).

Hashcash proofs of work are used in Bitcoin for block generation. In order for a block to be accepted by network participants, miners must complete a proof of work which covers all of the data in the block. The difficulty of this work is adjusted so as to limit the rate at which new blocks can be generated by the network to one every 10 minutes. Due to the very low probability of successful generation, this makes it unpredictable which worker computer in the network will be able to generate the next block.

For a block to be valid it must hash to a value less than the current target; this means that each block indicates that work has been done generating it. Each block contains the hash of the preceding block; thus, each block has a Chain of blocks that together contain a large amount of work. Changing a block (which can only be done by making a new block containing the same predecessor) requires regenerating all successors and redoing the work they contain. This protects the block chain from tampering. The most widely used proof-of-work scheme is based on SHA-256 and was introduced as a part of Bitcoin.

Drawbacks

Generating a specific hash value for the given transaction is resource consuming and time consuming. Thus, for generating a valid block after every transaction it requires huge amount of computer resources for generating the hash value. Even for validating the blockchain when a block is not valid will consume lots of resources and time as it would require hash value to be computed again for each block and checks its validity. Thus, everyone who is using the technology must have a workstation class computer.

4.2 PROPOSED SYSTEM

Since the existing system has the drawbacks of consuming lots of time and resources for generating a specific hash value and it takes immense time for validation of the blockchain if a block is not valid or if the data has been modified. Thus, in the proposed system for every data the hash value is computed and stored. Whenever the transaction and the storage process in the blockchain is over, the blockchain is written into a separate file and the hash value for that file is computed and stored. Thus, the integrity of the file is checked before a blockchain is added to it by generating the hash value for each blockchain files and comparing them. Thus, instead of comparing each hash value of the data stored in the blockchain we can compare the hash value of the file instead, thus saving time and computing resources.

In addition, the principles of blockchain is used in maintaining the integrity of the government transactions and in storage of sensitive data, where instead of handling the said processes by a single organization, we can decentralize the whole process where everyone who is involved in this have to contribute towards the integrity of the data stored in the blockchain. All the transactions and data stored is stored along with its hash value and whenever someone tries to illegally modify the data for their own benefits, we can identify the change and can stop it. Thus, there is a lot of transparency in the system. And thus, the corruption in the government over black money and money laundering can be stopped to an extent.

SYSTEM DESIGN

CHAPTER 5
SYSTEM DESIGN

5.1 System Architecture Diagram

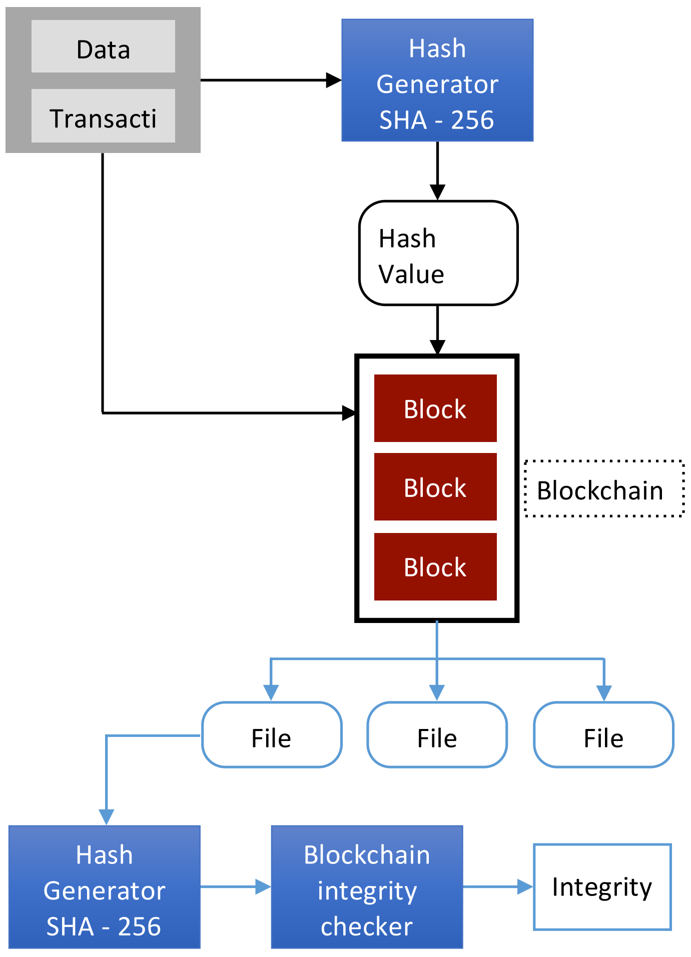


Fig 5.1.1

5.2 DATA FLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO.

DFD level 0

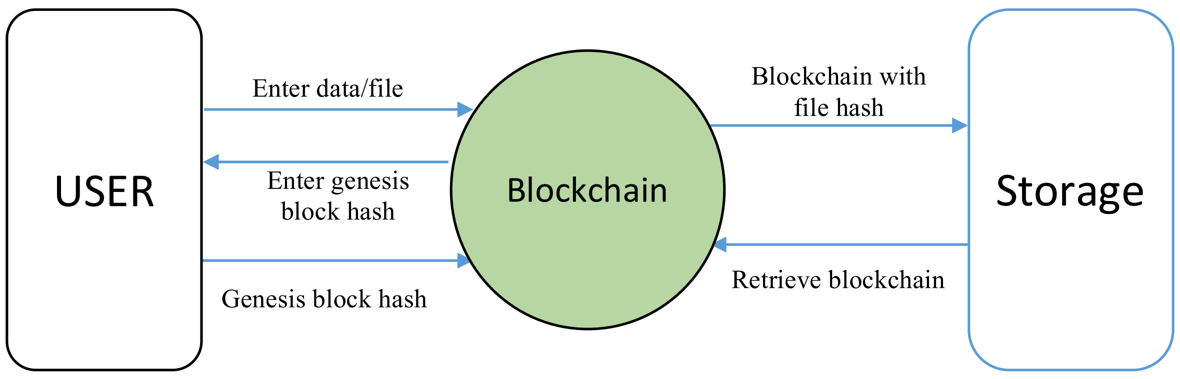


Fig 5.2.1

DFD level 1

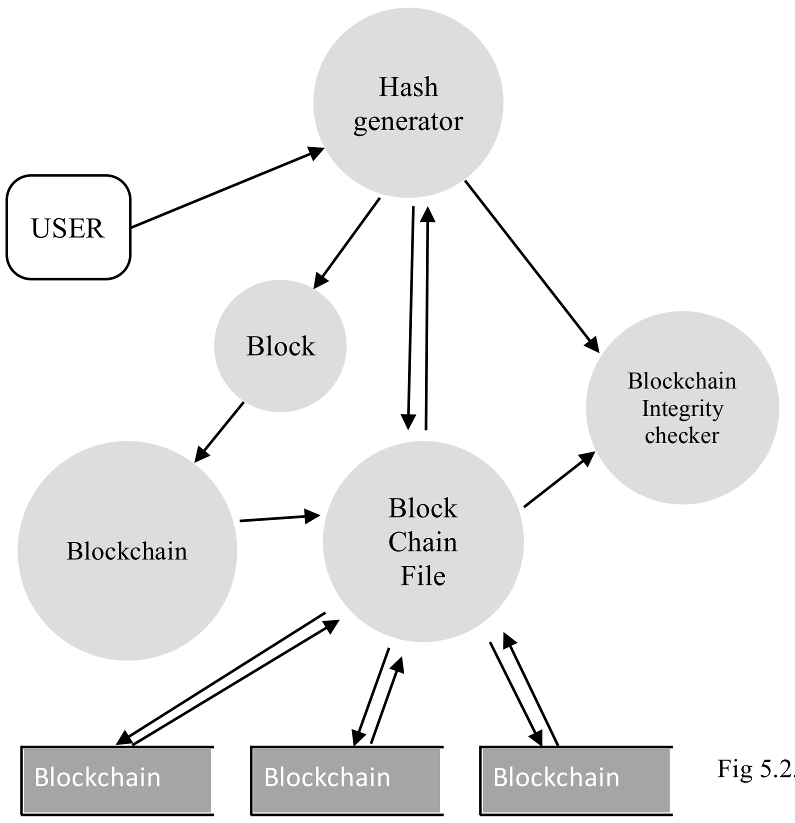


Fig 5.2.2

UML DIAGRAMS

CHAPTER 6
UML DIAGRAMS

The Unified Modelling Language (UML) was created to forge a common, semantically and syntactically rich visual modelling language for the architecture, design, and implementation of complex software systems both structurally and behaviorally. UML has applications beyond software development, such as process flow in manufacturing. It is analogous to the blueprints used in other fields and consists of different types of diagrams. In the aggregate, UML diagrams describe the boundary, structure, and the behavior of the system and the objects within it. UML is not a programming language but there are tools that can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design.

Types of UML Diagram

Class Diagram The most commonly used UML diagram, and the principal foundation of any object-oriented solution. Classes within a system, attributes and operations and the relationship between each class. Classes are grouped together to create class diagrams when diagramming large systems.

Component Diagram Displays the structural relationship of software system elements, most often employed when working with complex systems with multiple components. Components communicate using interfaces.

Deployment Diagram Illustrates system hardware and its software. Useful when a software solution is deployed across multiple machines with unique configurations.

Package Diagram There are two special types of dependencies defined between packages: package import and package merge. Packages can represent the different levels of a system to reveal the architecture. Package dependencies can be marked to show the communication mechanism between levels.

Activity Diagrams Graphically represented business or operational workflows to show the activity of any part or component in the system. Activity diagrams are used as an alternative to State Machine diagrams.

Sequence Diagram Shows how objects interact with each other and the order of occurrence. They represent interactions for a particular scenario.

State Machine Diagram Similar to activity diagrams, they describe the behavior of objects that behave in varying ways in their current state.

Use Case Diagram Represents a particular functionality of a system, created to illustrate how functionalities relate and their internal/external controllers (actors).

6.1 USE CASE DIAGRAM

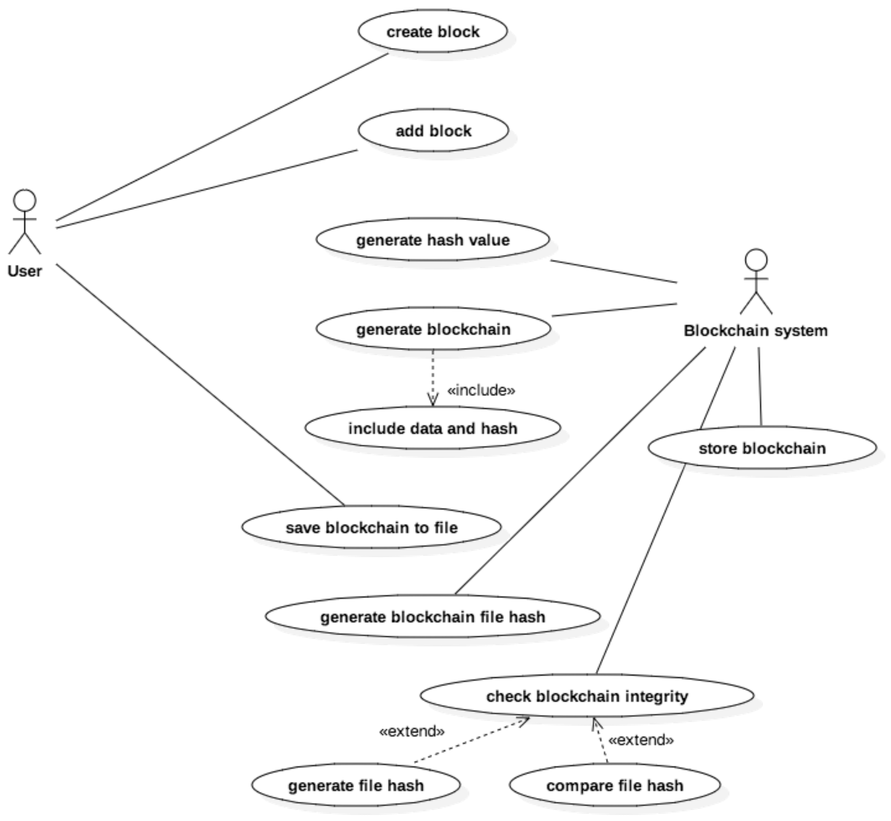


Fig 6.1.1

6.2 CLASS DIAGRAM

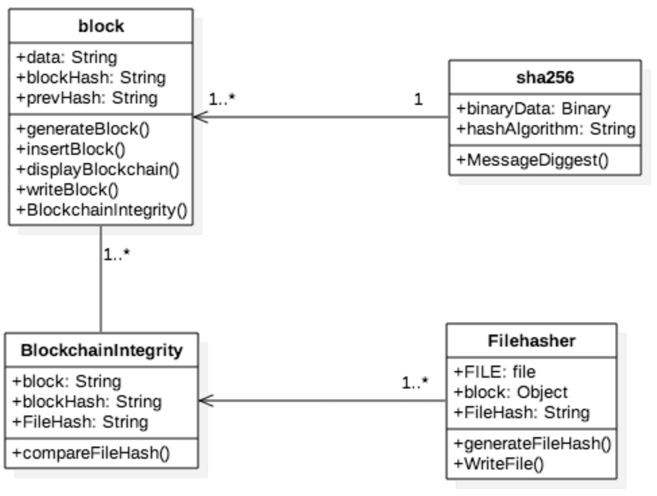


Fig 6.2.1

6.3 SEQUENCE DIAGRAM

Creation of blockchain

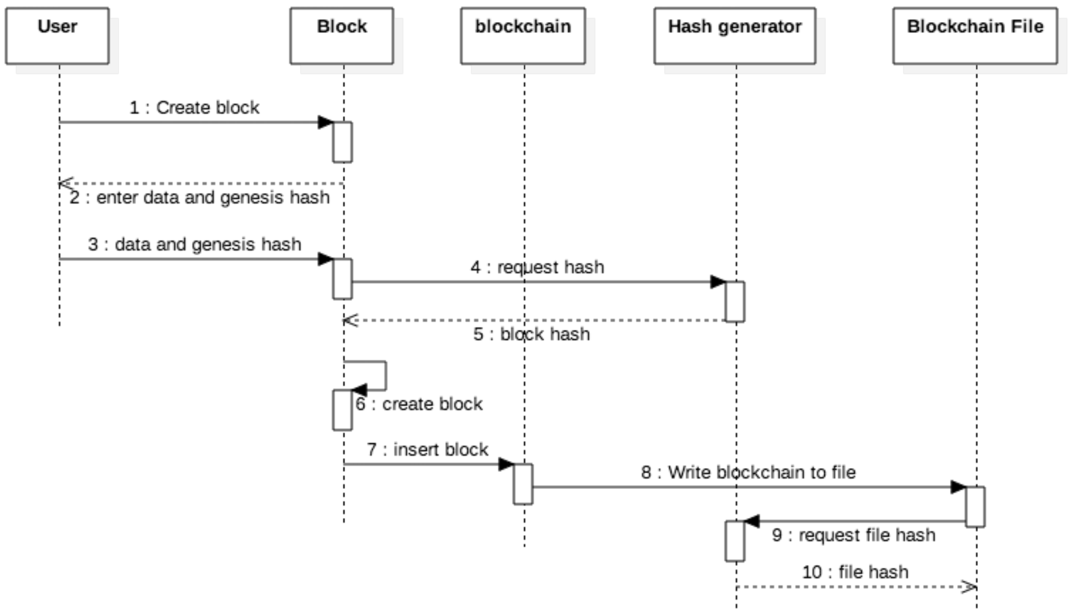


Fig 6.3.1

Blockchain Integrity

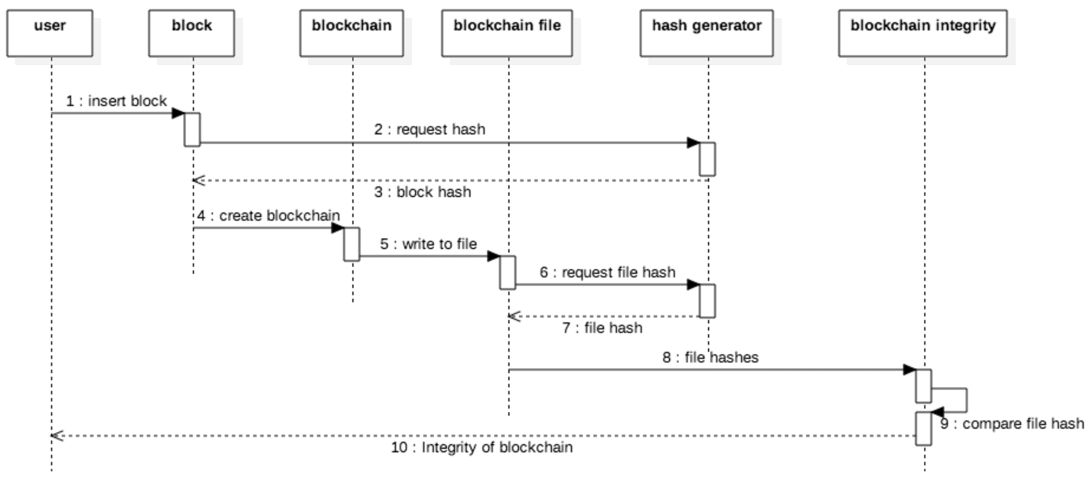


Fig 6.3.2

6.4 STATE DIAGRAM

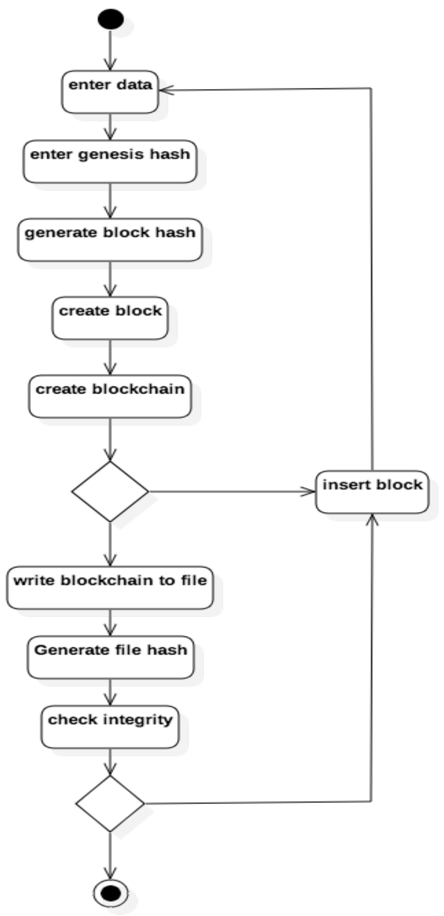


Fig 6.4.1

6.5 ACTIVITY DIAGRAM

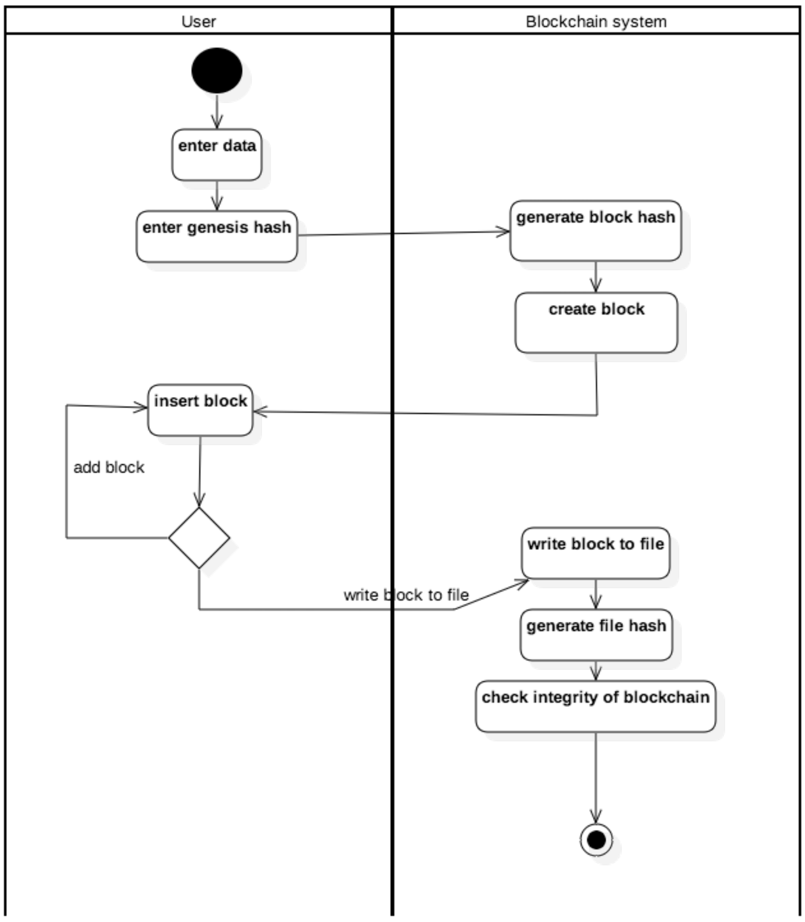


Fig 6.5.1

6.6 COMPONENT DIAGRAM

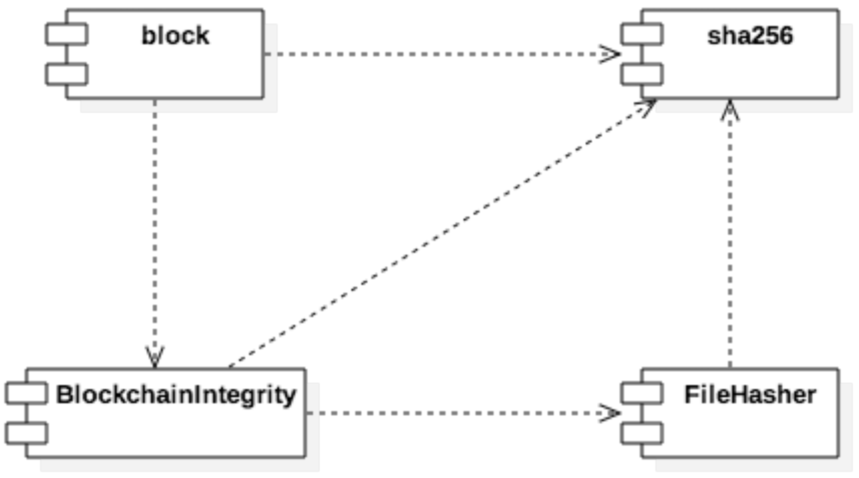


Fig 6.6.1

REQUIREMENT SPECIFICATION

CHAPTER 7
REQUIREMENT SPECIFICATION

Software requirements

- Operating System: Linux, Windows 7 & above
- Programming Language: JAVA - version 9
- IDE: Eclipse IDE for JAVA Developers
- Additional Packages: Google - gson

Hardware requirements

- Processor: Intel i3 & above
- RAM: 4 GB (Min)
- Storage: 500 GB (Min)

Eclipse

Eclipse is an integrated development environment (IDE) used in computer programming and is the most widely used Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins.

Google - gson

Gson is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java object. Gson can work with arbitrary Java objects including pre-existing objects that you do not have source-code of. There are a few open-source projects that can convert Java objects to JSON. However, most of them require that you place Java annotations in your classes; something that you cannot do if you do not have access to the source-code. Most also do not fully support the use of Java Generics. Gson considers both of these as very important design goals.

SYSTEM IMPLEMENTATION

CHAPTER 8
SYSTEM IMPLEMENTATION

This project is implemented completely in java. The implementation is done through six modules. The modules are as follows:

1. mainBlock
2. block
3. fileHasher
4. blockchainIntegrity
5. getFileHash
6. SHA-256

Each module mentioned above is described individually with their code and input and output data.

8.1 mainBlock

This module is responsible for the creation of blocks and blockchain, displaying blockchain and writing the blockchain to the required file and checking the validity of the blockchain. The input for this module is user data and genesis block hash at the creation of first block. The output of this module is blockchain represented in a JSON (JavaScript Object Notation) format for ease in sharing of data between multiple systems.

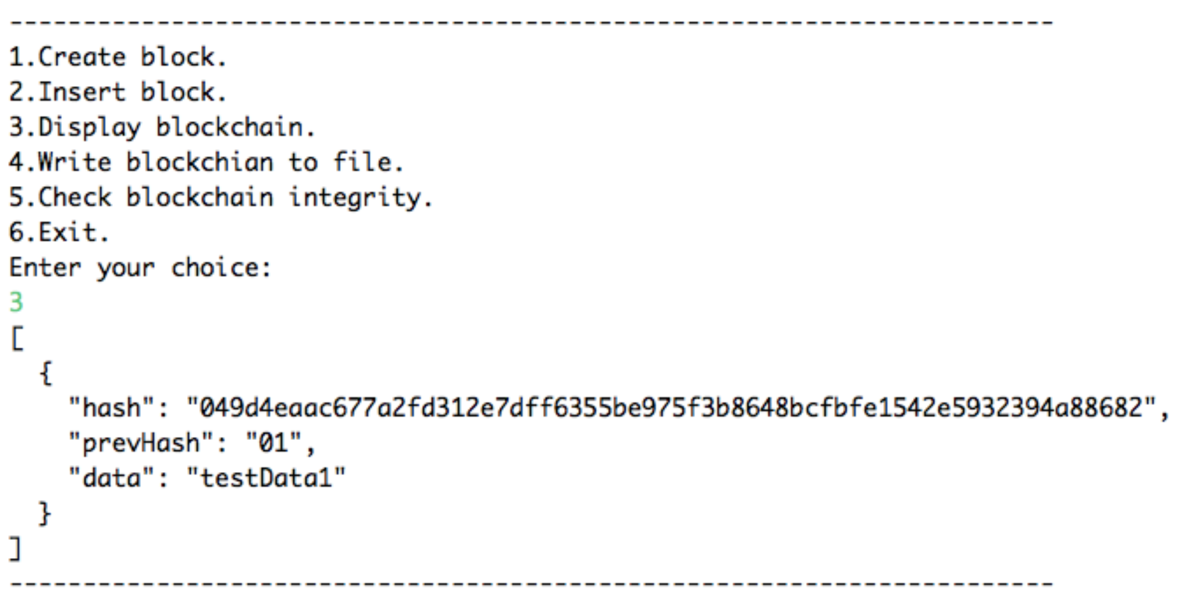
Source code:

```
package BlockChain;
import java.util.ArrayList;
import com.google.gson.GsonBuilder;
import java.util.Scanner;
import java.io.IOException;
import java.security.NoSuchAlgorithmException;
```

```
public class mainBlock {

        public static ArrayList<Block> blockchain = new ArrayList<Block>();
        public static int difficulty = 1;
        public static void main(String[] args) throws
        NoSuchAlgorithmException, IOException {
                Scanner sc = new Scanner(System.in);
                String Data = new String();
                String genesisBlockHash = new String();
                int ch = 0;
                fileHasher FH = new fileHasher();
                do {
                System.out.println("--------------------------------------------------");
                System.out.println("1.Create block.");
                System.out.println("2.Insert block.");
                System.out.println("3.Display blockchain.");
                System.out.println("4.Write blockchian to file.");
                System.out.println("5.Check blockchain integrity.");
                System.out.println("6.Exit.");
                System.out.println("Enter your choice:");
                ch = sc.nextInt();
                switch(ch) {
                        case 1: System.out.println("Enter block data:");
                                Data = sc.next();
                                System.out.println("Enter genesis block hash:");
                                genesisBlockHash = sc.next();
                                blockchain.add(new Block(Data, genesisBlockHash));
                                //blockchain.get(0).mineBlock(difficulty);
                                break;
```

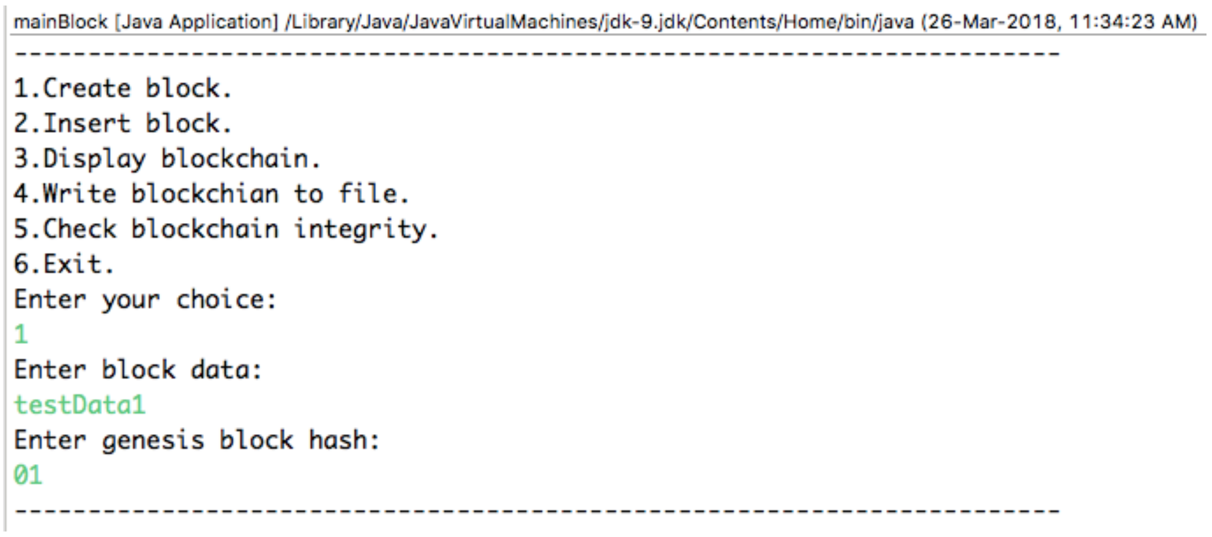
```
case 2: System.out.println("Enter block data:");
          Data = sc.next();
          blockchain.add(new Block(Data,
          blockchain.get(blockchain.size()1).hash));
          break;
case 3:
          String blockchainJson = new
          GsonBuilder().setPrettyPrinting()
                    .create().toJson(blockchain);
          System.out.println(blockchainJson);
          break;
case 4: String blockChain = new
          GsonBuilder().setPrettyPrinting().create().
                    toJson(blockchain);
          try {
                              FH.blockFileWriter(blockChain);
                              String blockchainFileHash =
                                                  FH.retFileHash();
                              System.out.println("Blockchain File
                                        Hash : "+blockchainFileHash);
                              } catch (IOException e) {
                                        e.printStackTrace();
                              }
                    break;
case 5: blockchainIntegrity bc = new blockchainIntegrity();
            bc.BlockChainIntegrity();
                    break;
                    case 6: break;
                        }
        System.out.println("-----------------------------------------------------");
        } while(ch != 6);
        sc.close();
}
public static Boolean chainValidity() throws NoSuchAlgorithmException {
        Block currentBlock;
        Block prevBlock;
        for(int i = 1; i < blockchain.size(); i++)
        {
                currentBlock = blockchain.get(i);
                prevBlock = blockchain.get(i-1);
                if(!currentBlock.hash.equals(currentBlock.blockHASH()))
                {
                        System.out.println("Current hash is not equal..");
                        return false;
                }
                if(!prevBlock.hash.equals(currentBlock.prevHash))
                {
                        System.out.println("Previous hash is not equal...");
                        return false;
                }
        }
        return true;
        }
}
```

Screenshot



```
-----------------------------------------------------------------------
1.Create block.
2.Insert block.
3.Display blockchain.
4.Write blockchian to file.
5.Check blockchain integrity.
6.Exit.
Enter your choice:
3
[
  {
    "hash": "049d4eaac677a2fd312e7dff6355be975f3b8648bcfbfe1542e5932394a88682",
    "prevHash": "01",
    "data": "testData1"
  }
]
-----------------------------------------------------------------------
```

Fig 8.1.1



```
mainBlock [Java Application] /Library/Java/JavaVirtualMachines/jdk-9.jdk/Contents/Home/bin/java (26-Mar-2018, 11:34:23 AM)
-----------------------------------------------------------------------
1.Create block.
2.Insert block.
3.Display blockchain.
4.Write blockchian to file.
5.Check blockchain integrity.
6.Exit.
Enter your choice:
1
Enter block data:
testData1
Enter genesis block hash:
01
-----------------------------------------------------------------------
```

Fig 8.1.2

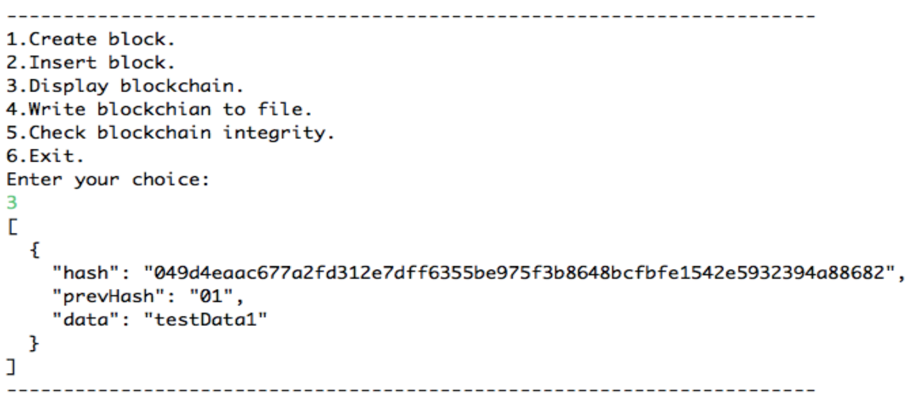
8.2 block

This module describes how the content of the block is laid. It is responsible for creation of block and blockchain. The **input** for this module is **user data, hash and prevHash**(hash value of the previous block. The **output** for this module is the block itself which has the user data, hash and prevHash in it.

Source code

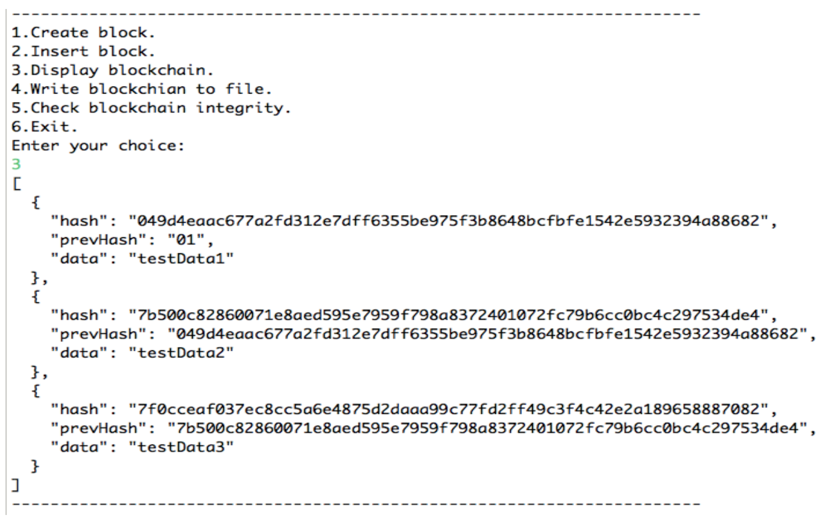
```
package BlockChain;
import java.security.NoSuchAlgorithmException;
//import java.util.Date;
public class Block {
        public String hash;
        public String prevHash;
        private String data;
        public Block(String data, String prevHash) throws
        NoSuchAlgorithmException{
                this.data = data;
                this.prevHash = prevHash;
                //this.timeStamp = new Date().getTime();
                this.hash = blockHASH();
        }
        String blockHASH() throws NoSuchAlgorithmException{
                String calculatedBlockHash;
                SHA_256 sh = new SHA_256();
                calculatedBlockHash = sh.retSHA(prevHash+data);
                return calculatedBlockHash;
        }
        public void displayBlockHash()
        {
                System.out.println("Block Data :"+data);
                System.out.println("Block HASH:"+hash);
                System.out.println("Block PREV_HASH:"+prevHash);
                //System.out.println("Time Stamp : "+timeStamp);
        }
}
```

Screenshot



```
----------------------------------------------------------------------
1.Create block.
2.Insert block.
3.Display blockchain.
4.Write blockchian to file.
5.Check blockchain integrity.
6.Exit.
Enter your choice:
3
[
  {
    "hash": "049d4eaac677a2fd312e7dff6355be975f3b8648bcfbfe1542e5932394a88682",
    "prevHash": "01",
    "data": "testData1"
  }
]
----------------------------------------------------------------------
```

Fig 8.2.1



```
----------------------------------------------------------------------
1.Create block.
2.Insert block.
3.Display blockchain.
4.Write blockchian to file.
5.Check blockchain integrity.
6.Exit.
Enter your choice:
3
[
  {
    "hash": "049d4eaac677a2fd312e7dff6355be975f3b8648bcfbfe1542e5932394a88682",
    "prevHash": "01",
    "data": "testData1"
  },
  {
    "hash": "7b500c82860071e8aed595e7959f798a8372401072fc79b6cc0bc4c297534de4",
    "prevHash": "049d4eaac677a2fd312e7dff6355be975f3b8648bcfbfe1542e5932394a88682",
    "data": "testData2"
  },
  {
    "hash": "7f0cceaf037ec8cc5a6e4875d2daaa99c77fd2ff49c3f4c42e2a189658887082",
    "prevHash": "7b500c82860071e8aed595e7959f798a8372401072fc79b6cc0bc4c297534de4",
    "data": "testData3"
  }
]
----------------------------------------------------------------------
```

Fig 8.2.2

8.3 fileHasher

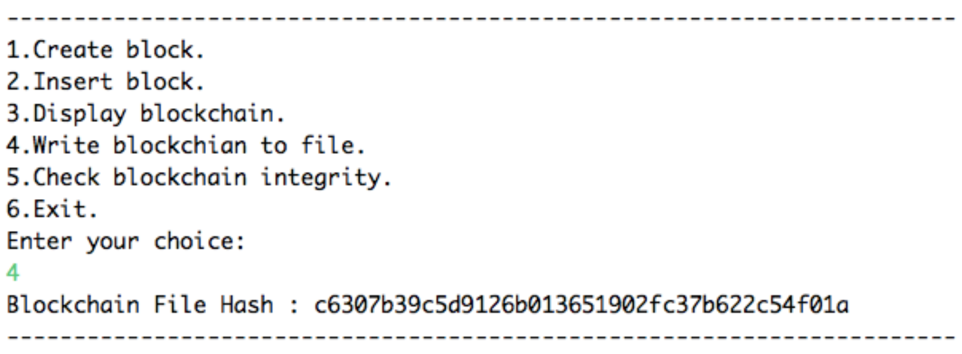
This module is responsible for writing the blockchain into the required number of files and get the hash value of each file whenever the blockchain is written into the file. The **input** for this module is **the file to which the blockchain should be written.** The **output** for this module is **hash value for the file.**

Source code

```
package BlockChain;
import java.io.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
public class fileHasher {

        public String retFileHash() throws IOException{
                getFileHash hs = new getFileHash();
                String rethash = new String();
                File file = new File("/Users/prasannamanikandan/Documents/
                Eclipse workspace/File_hasher/src/File_hasher/blockchain1.txt");
                try {
                        MessageDigest shaDigest =
                                MessageDigest.getInstance("SHA-1");
                        String shaChecksum = hs.getFileHashval(shaDigest, file);
                        rethash = shaChecksum;
                } catch (NoSuchAlgorithmException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                return rethash;
        }
```

```
	public void blockFileWriter(String s) throws IOException {
		File file = new File("/Users/prasannamanikandan/Documents/
		Eclipse workspace/File_hasher/src/File_hasher/blockchain1.txt");
		File file2 = new File("/Users/prasannamanikandan/Documents/
		Eclipse workspace/File_hasher/src/File_hasher/blockchain2.txt");
		//File  file3  =  new  File("/Users/prasannamanikandan/Documents/
		Eclipse workspace/File_hasher/src/File_hasher/blockchain3.txt");
		FileWriter fileWriter = new FileWriter(file.getAbsoluteFile(),true);
		FileWriter fileWriter2 = new
				FileWriter(file2.getAbsoluteFile(),true);
		//FileWriter fileWriter3 = new
				FileWriter(file3.getAbsoluteFile(),true);
		fileWriter.write(s);
		fileWriter.flush();
		fileWriter.close();

		fileWriter2.write(s);
		fileWriter2.flush();
		fileWriter2.close();

		//fileWriter3.write(s);
		//fileWriter3.flush();
		//fileWriter3.close();
	}
}
```

Screenshot



```
-----------------------------------------------------------------------
1.Create block.
2.Insert block.
3.Display blockchain.
4.Write blockchian to file.
5.Check blockchain integrity.
6.Exit.
Enter your choice:
4
Blockchain File Hash : c6307b39c5d9126b013651902fc37b622c54f01a
-----------------------------------------------------------------------
```

Fig 8.3.1

8.4 blockchainIntegrity

This module is responsible for checking the integrity of the blockchain by comparing all the hash values of each blockchain file and checking whether all the hash values are same or not. The **input** for this module is the **hash values** of each file and the **output** for this module is statement stating whether the blockchain is valid or not.

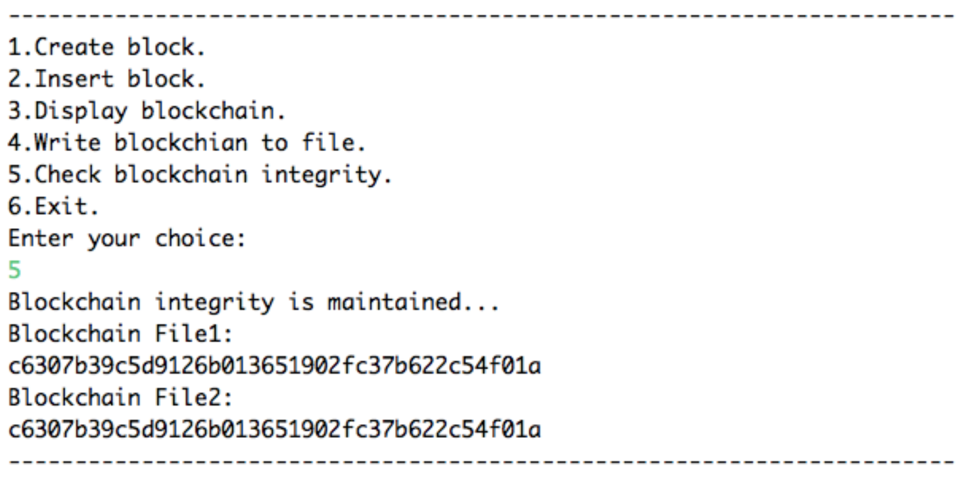
Source code

```
package BlockChain;
import java.io.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class blockchainIntegrity {
        public void BlockChainIntegrity() throws IOException {
        getFileHash hs1 = new getFileHash();
        File file1 = new File("/Users/prasannamanikandan/Documents/
                Eclipse workspace/File_hasher/src/File_hasher/blockchain1.txt");
                File file2 = new File("/Users/prasannamanikandan/Documents/
                Eclipse workspace/File_hasher/src/File_hasher/blockchain2.txt");
                try {
```

```
                MessageDigest shaDigest =
                        MessageDigest.getInstance("SHA-1");
                String filehs1 = hs1.getFileHashval(shaDigest, file1);
                String filehs2 = hs1.getFileHashval(shaDigest, file2);
                if(filehs1.equals(filehs2))
                {
                        System.out.println("Blockchain integrity is
                                                                maintained...");
                        System.out.println(filehs1);
                        System.out.println(filehs2);
                }
                else {
                        System.out.println("Blockchain is not valid...");
                        System.out.println(filehs1);
                        System.out.println(filehs2);
                }
        } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
        }
    }
}
```

Screenshot

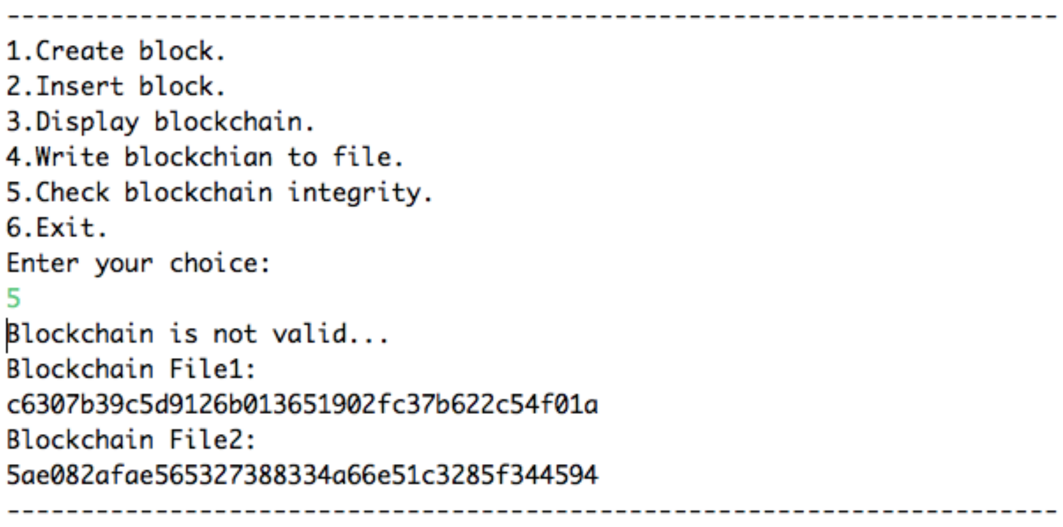
Valid Blockchain:



```
----------------------------------------------------------------------
1.Create block.
2.Insert block.
3.Display blockchain.
4.Write blockchian to file.
5.Check blockchain integrity.
6.Exit.
Enter your choice:
5
Blockchain integrity is maintained...
Blockchain File1:
c6307b39c5d9126b013651902fc37b622c54f01a
Blockchain File2:
c6307b39c5d9126b013651902fc37b622c54f01a
----------------------------------------------------------------------
```

Fig 8.4.1

Invalid Blockchain:



```
----------------------------------------------------------------------
1.Create block.
2.Insert block.
3.Display blockchain.
4.Write blockchian to file.
5.Check blockchain integrity.
6.Exit.
Enter your choice:
5
Blockchain is not valid...
Blockchain File1:
c6307b39c5d9126b013651902fc37b622c54f01a
Blockchain File2:
5ae082afae565327388334a66e51c3285f344594
----------------------------------------------------------------------
```

Fig 8.4.2

8.5 getFileHash

This module is responsible for generating hash value for a given file using MessageDigest class. The **input** for this module is the **blockchain file** and the **output** for this module is the **hash value for the file.**

Source code

```
package BlockChain;
import java.security.MessageDigest;
import java.io.*;
public class getFileHash {
        public String getFileHashval(MessageDigest digest, File file)throws
                                                                IOException {
                FileInputStream fis = new FileInputStream(file);
                byte[] byteArray = new byte[1024];
                int bytesCount = 0;
                while ((bytesCount = fis.read(byteArray)) != -1) {
                        digest.update(byteArray, 0, bytesCount);};
                fis.close();
                byte[] bytes = digest.digest();
                StringBuilder sb = new StringBuilder();
                for(int i=0; i< bytes.length ;i++)
                {
                        sb.append(Integer.toString((bytes[i] & 0xff) + 0x100,
                                16).substring(1));
                }
           return sb.toString();
        }
}
```

8.6 SHA_256

This module is responsible for generating hash value for the given data. The **input** for this module is **user data** and the output for this module is the **hash value** for the given data.

Source code

```
package BlockChain;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.nio.charset.StandardCharsets;
import java.math.BigInteger;
public class SHA_256 {
        String retSHA(String data) throws NoSuchAlgorithmException{
        MessageDigest md = MessageDigest.getInstance( "SHA-256" );
        md.update( data.getBytes( StandardCharsets.UTF_8 ) );
        byte[] digest = md.digest();
        String hex = String.format( "%064x", new BigInteger( 1, digest ) );
        return hex;
        }
}
```

TESTING AND MAINTENANCE

CHAPTER 9
TESTING AND MAINTENANCE

Software testing is an investigation conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use.

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test

- Meets the requirements that guided its design and development,
- Responds correctly to all kinds of inputs,
- Performs its functions within an acceptable time,
- Is sufficiently usable,
- Can be installed and run in its intended environments, and
- Achieves the general result its stakeholders desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job of testing is an iterative process as when one bug is fixed, it can illuminate other, deeper bugs, or can even create new ones. Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors. Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under an agile approach, requirements, programming, and testing are often done concurrently.

9.1 Levels of Testing

- Functional testing
- Unit testing
- Integration testing
- System testing
- Regression testing
- Acceptance testing
- Alpha testing
- Beta testing
- Nonfunctional testing

Functional Testing

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for.

Unit testing

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

Integration testing

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

System testing

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialised testing team.

Regression testing

Whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by this change. Regression testing is performed to verify that a fixed bug hasn't resulted in another functionality or business rule violation. The intent of regression testing is to ensure that a change, such as a bug fix should not result in another fault being uncovered in the application.

Acceptance testing

This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client’s requirement. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application. By performing acceptance tests on an application, the testing team will reduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system.

9.2 Unit testing - testing blockchain file integrity

9.2.1 Blockchain Integrity checker:

Code to be tested:

```
package blockchainTesting;
import java.security.MessageDigest;
import java.io.*;
import java.security.NoSuchAlgorithmException;

public class blockchainFileHash {
        public String getFileHashval(MessageDigest digest, File file)throws
IOException {
                FileInputStream fis = new FileInputStream(file);
                byte[] byteArray = new byte[1024];
                int bytesCount = 0;
                while ((bytesCount = fis.read(byteArray)) != -1) {
                digest.update(byteArray, 0, bytesCount);
                };
                fis.close();
                byte[] bytes = digest.digest();
                StringBuilder sb = new StringBuilder();
                for(int i=0; i< bytes.length ;i++)
                {
                        sb.append(Integer.toString((bytes[i] & 0xff) + 0x100,
                        16).substring(1));
                }
            return sb.toString();
        }
        public String getFileHash() throws IOException {
                String fileHashVal = new String();
                File file = new
                File("/Users/prasannamanikandan/Documents/Eclipse
                workspace/File_hasher/src/File_hasher/blockchain1.txt");
                try {
                        MessageDigest shaDigest =
                        MessageDigest.getInstance("SHA-1");
                        blockchainFileHash bch = new blockchainFileHash();
                        String shaCheckSum = bch.getFileHashval(shaDigest, file);
                        fileHashVal = shaCheckSum;
                }
                catch (NoSuchAlgorithmException e) {
                        e.printStackTrace();
                }
                return fileHashVal;
        }
}
```

Testing code:

```
package blockchainTesting;
import static org.junit.jupiter.api.Assertions.*;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import org.junit.jupiter.api.Test;
import static org.junit.Assert.assertEquals;
public class blockchainTestUnit {
```

```
String filehash1 = new String();
String filehash2 = new String();
public String getFileHashval(MessageDigest digest, File file)throws
IOException {
        FileInputStream fis = new FileInputStream(file);
        byte[] byteArray = new byte[1024];
        int bytesCount = 0;
        while ((bytesCount = fis.read(byteArray)) != -1) {
              digest.update(byteArray, 0, bytesCount);
        };
        fis.close();
        byte[] bytes = digest.digest();
        StringBuilder sb = new StringBuilder();
        for(int i=0; i< bytes.length ;i++)
        {
          sb.append(Integer.toString((bytes[i] & 0xff) + 0x100,16).substring(1));
         }
           return sb.toString();
        }
        public String getFileHashval() throws IOException{
                String fileHashVal = new String();
                File file = new
                File("/Users/prasannamanikandan/Documents/Eclipse
                workspace/File_hasher/src/File_hasher/blockchain2.txt");
                try {
                MessageDigest shaDigest = MessageDigest.getInstance("SHA-1");
                        blockchainTestUnit bch = new blockchainTestUnit();
                        String shaCheckSum = bch.getFileHashval(shaDigest, file);
                        fileHashVal = shaCheckSum;
                filehash1 = shaCheckSum;
            }
            catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
            blockchainFileHash block = new blockchainFileHash();
            filehash2 = block.getFileHash();
            return fileHashVal;
        }
        @Test
        void test() throws IOException {
                blockchainTestUnit b1 = new blockchainTestUnit();
                blockchainFileHash b2 = new blockchainFileHash();
                assertEquals(b1.getFileHashval(),b2.getFileHash());
        }
}
```

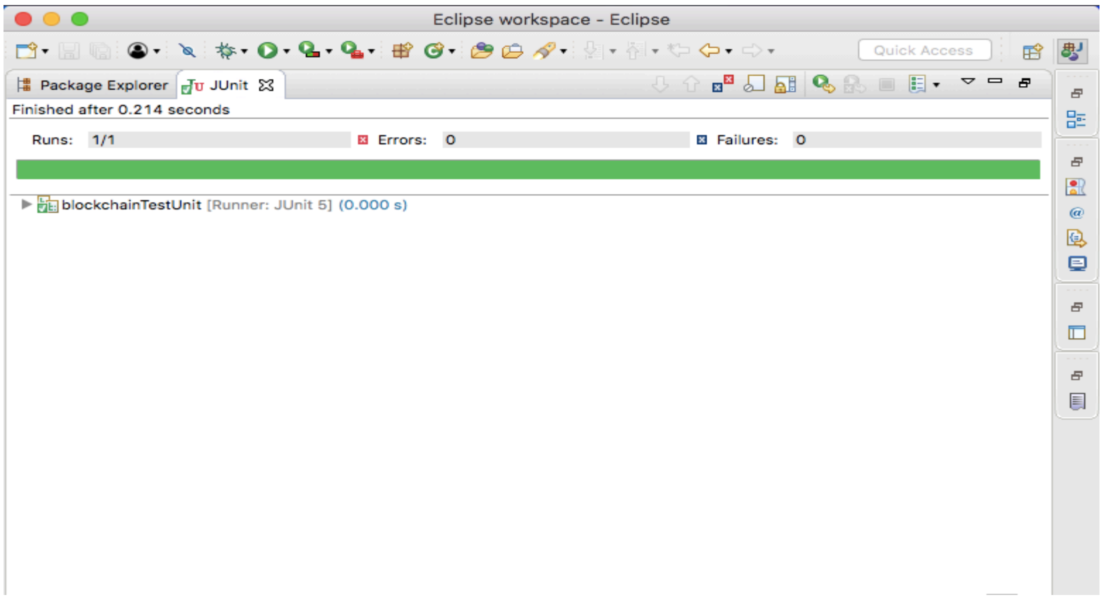
Test case 1:

Testing the integrity of two blockchain file by comparing its hash value, without modifying its content.

Input: Two blockchain file and its hash value

Expected output: Blockchain file integrity maintained

Test result: Pass



```
Finished after 0.214 seconds
Runs: 1/1      Errors: 0      Failures: 0
▶ blockchainTestUnit [Runner: JUnit 5] (0.000 s)
```

Fig 9.2.1.1

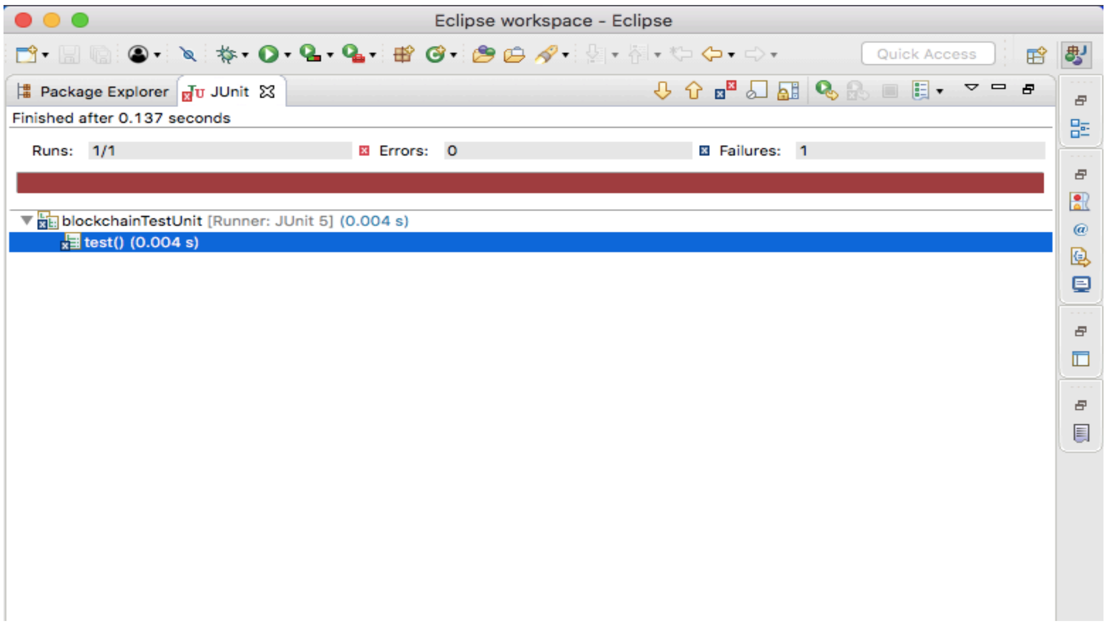
Test case 2:

Testing the integrity of two blockchain file by comparing its hash value, by modifying content of one of the blockchain file.

Input: Two blockchain file (one modified) and its hash value

Expected output: Blockchain file integrity maintained

Test result: Fail



```
Finished after 0.137 seconds
Runs: 1/1      Errors: 0      Failures: 1
▼ blockchainTestUnit [Runner: JUnit 5] (0.004 s)
    test() (0.004 s)
```

Fig 9.2.1.2

9.2.2 SHA-256

Code to be tested:

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.nio.charset.StandardCharsets;
import java.math.BigInteger;

public class sha256_class {
        String retSHA(String data) throws NoSuchAlgorithmException{
                MessageDigest md = MessageDigest.getInstance( "SHA-256" );
                md.update( data.getBytes( StandardCharsets.UTF_8 ) );
           byte[] digest = md.digest();
           String hex = String.format( "%064x", new BigInteger( 1, digest ) );
           return hex;
        }
}
```

Testing Code:

```
import static org.junit.jupiter.api.Assertions.*;
import java.io.IOException;
import java.security.NoSuchAlgorithmException;
import java.util.*;
import org.junit.jupiter.api.Test;

class hashCompare_test {
        public String hashCompare() throws NoSuchAlgorithmException{
                sha256_class sha = new sha256_class();
                Scanner sc = new Scanner(System.in);
                String data1,hash1;
```

```
            System.out.println("Enter the data to be hashed:");
            data1 = sc.next();
            hash1 = sha.retSHA(data1);
            sc.close();
            return hash1;
            }

    @Test
    void test() throws NoSuchAlgorithmException{
            String hash2 = "test";
            sha256_class sh = new sha256_class();
            String hashcomp = sh.retSHA(hash2);
            hashCompare_test hc = new hashCompare_test();
            String hashcomp2 = hc.hashCompare();
            assertEquals(hashcomp,hashcomp2);
    }

}
```

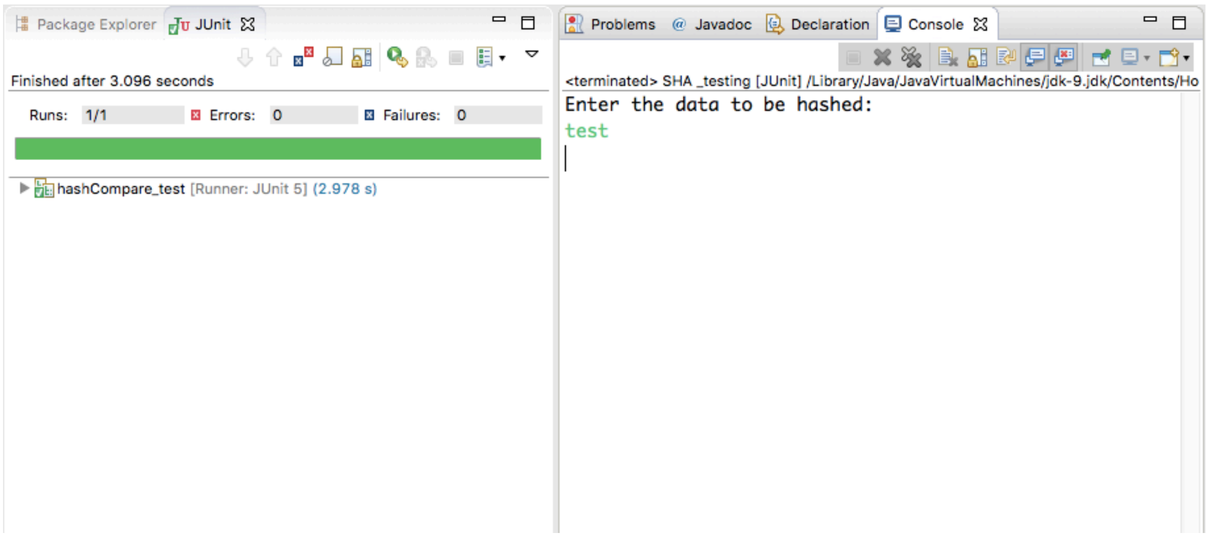
Test Case 3:

Testing whether the hash value is generated uniquely or not and comparing the hash value with test data.

Input: data

Expected Output: Same hash value

Result: Pass



```
Finished after 3.096 seconds
Runs: 1/1    Errors: 0    Failures: 0
hashCompare_test [Runner: JUnit 5] (2.978 s)
```

```
<terminated> SHA _testing [JUnit] /Library/Java/JavaVirtualMachines/jdk-9.jdk/Contents/Ho
Enter the data to be hashed:
test
```

Fig 9.2.2.1

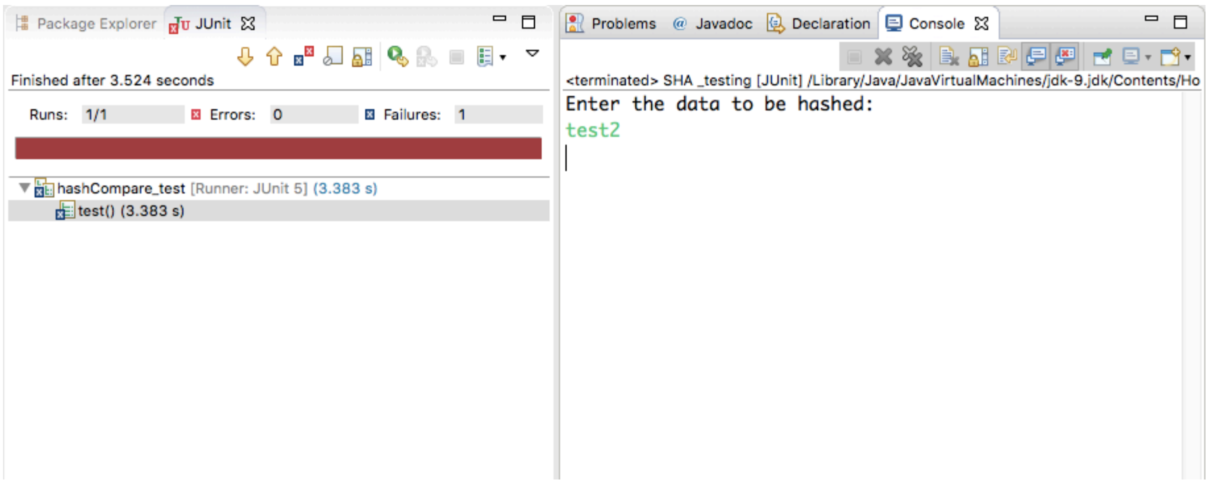
Test case 4:

Testing whether the hash value is generated correctly and comparing it with test data.

Input: data (different than test data).

Expected output: Same hash value.

Result: fail



```
Finished after 3.524 seconds
Runs: 1/1    Errors: 0    Failures: 1
hashCompare_test [Runner: JUnit 5] (3.383 s)
    test() (3.383 s)
```

```
<terminated> SHA _testing [JUnit] /Library/Java/JavaVirtualMachines/jdk-9.jdk/Contents/Ho
Enter the data to be hashed:
test2
```

Fig 9.2.2.2

9.2.1 Test Cases

Test Case ID	Test Item	Input Specification	Output Specification	Pass/Fail
Test case 1	Blockchain integrity checker	Two blockchain files	Blockchain integrity maintained	Pass
Test case 2	Blockchain integrity checker	Two blockchain files with one modified	Blockchain integrity not maintained	Fail
Test case 3	SHA256 hashing	User data and test data	Same hash value	Pass
Test case 4	SHA256 hashing	User data and test data(modified)	Hash value is different	Fail

9.3 MAINTENANCE

After a software system has been verified, tested and implemented, it must continue to be maintained. Maintenance routines will vary depending on the type and complexity of the technology. Many software systems will come with a maintenance schedule or program recommended by the developer. Maintenance could be provided by the developer as part of the purchase agreement for the technology.

Systems will need to be maintained to ensure that they continue to perform to the level demonstrated during the system testing stage. If systems deteriorate, there is a risk that the systems will not perform to the required standard.

Ongoing monitoring or testing systems may be installed to ensure that maintenance needs are identified and met where necessary. Where systems are in long-term use, a system can be designed to monitor feedback from users and conduct any modifications or maintenance as needed. Where modifications to software are made as a result of system maintenance or upgrades, it may be necessary to instigate further rounds of system verification and testing to ensure that standards are still met by the modified system.

CONCLUSION AND FUTURE ENHANCEMENT

CHAPTER 10
CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

Thus in this project we have studied about blockchain technology, its usage and its implementation. It is used majorly in Cryptocurrency, which involves in generating specific hash value for proof of work. This demands high computational power. As for this project the blockchain technology is used in conducting and storing government transactions so that it cannot be modified illegally to reduce corruption. This projects provides the methods for storing data in blockchain as a file and maintaining its integrity by using the hash value generated for each file. By this method we can stop government corruption to an extent.

Future Enhancement

- Enhance the security of the system
- Support for storing multiple types of data and files
- Support for multiple storage technology
- Further improve the resources and time required for computing hash value

REFERENCES

11. REFERENCES

1. https://blockgeeks.com/guides/what-is-blockchain-technology/
2. https://en.wikipedia.org/wiki/Blockchain
3. https://www.coindesk.com/information/what-is-blockchain-technology/
4. https://www.ibm.com/blockchain/what-is-blockchain.html
5. https://medium.com/programmers-blockchain/create-simple-blockchain-java
6. Blockchain Technology as an enabler of service (Aug 2017) - Stefen Seebacher
7. Using Blockchain Technology to Validate the Integrity and Confidentiality of Backups Versions on the Cloud (Feb 2018) – Badr Aleidi, Abdulaziz Albesher
8. A review of blockchain (Nov 2016) – George Pirlea
9. Blockchain platform with proof-of-work based on analog Hamiltonian optimizers (Feb 2018) – K. P. Kalinin

APPENDIX

APPENDIX
(PUBLICATION DETAILS)

Paper Title	Decentralized Applications using Blockchain – Managing government corruption
Authors	Mr. S Gopi, M Prasanna, R Shijil, S M Aslam, M Prakash Kumar
Journal Name	International Research Journal of Engineering and Technology (IRJET)
Edition	Volume 5 Issue 4, April 2018
Month and Year	April 2018