



De-Centralized Secure Transparent Systems to Manage Financial Transactions

A dissertation submitted for the Degree of Master of
Science in Information Security

Kaludewa Chamil Lakmal De Silva
University of Colombo School of Computing
Sri Lanka
2019



DECLARATION

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute. To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: K.C.L. De Silva

Signature:

Date:.....

This is to certify that this thesis is based on the work of Mr. K.C.L. De Silva Under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by: Supervisor Name: Dr. Kasun de Zoysa

Signature:

Date:.....

Mr.Kenneth Manjula Thilakarathna
MSc. Research Project Advisor, UCSC

Signature:

Date:.....

ACKNOWLEDGMENT

The "De-Centralized Secure Transparent Systems to Manage Financial Transactions" was carried out as a part of the Master of Science in Information Security program at University of Colombo School of Computing.

I would like to thank all who helped me in numerous ways to make this work a success. Especially I am truly grateful for the valuable guidance and assistance given by my supervisor, Kasun de Zoysa, Senior Lecturer at University of Colombo School of Computing. and also Mr.Kenneth Manjula Thilakarathna was a great Mr.Kenneth Manjula Thilakarathna was a great mentor, visiting Lecturer at University of Colombo School of Computing. Also I would like to thank Dr.Manjusri Wickramasinghe and rest of the staff members at UCSC for some expertise help, providing advice, constant constructive criticism of my ideas and their valuable comments throughout this work.

I am grateful to my friends and family for the maximum support and courage given during the time to make this a success.

ABSTRACT

De-Centralized Secure Transparent Systems to Manage Financial Transactions

By

K.C.L. De Silva, chamil.lakmal@gmail.com
UCSC, Colombo 07.

The projects run by the Sri Lankan Government has a tendency to overspend due to fraudulent transactions and abuse of power. This is a huge problem in Sri Lanka since this affects the finances of the country and hinders the development of the country as a whole. In worst case scenarios these projects completely fail resulting in a tremendous wastage of resources. With a very limited number of stakeholders having access to financial information on such projects makes it easier for this type of fraudulent activity. Even though there is a current system in place to audit transactions the current system is easy to tamper with. As a result stakeholders committing fraud has the ability to escape any consequences and repeat the actions in different projects.

With blockchain technology transactions can be recorded in a way that is not permeable leaving no room for tempering and deleting information from financial records. With the use of digital signatures the identity of the person who did the transaction will not be an issue. The process of auditing will take place before the transaction is authorized minimizing the chances of fraudulent transactions. The blockchain by definition is a public ledger system therefore the transparency of all financial transactions are guaranteed. The researcher believes that integration of blockchain to manage financial transactions for projects run by the Sri Lankan Government will reduce corruption and fraud as well as make these projects cost efficient, improving their chances of success and help achieve developmental goals of the Sri Lankan Government.

The proposed system will take the Sri Lankan legal frame work into consideration as well as the requirements of the auditing process for government projects. All transactions need to be approved by all relevant stakeholders ensuring that the proposed transaction will not exceed the amount required. More over the money will directly be transferred to the wallet of the stakeholder requiring the financial transaction avoiding intermediaries from cashing in. After the transaction is authenticated by all relevant stakeholders smart contracts in the blockchain will carry out the transaction automatically. Digital signatures will assure that the no stakeholder can deny receiving the money, authorizing a transaction or requesting a transaction.

Contents

TABLE OF CONTENTS	v
LIST OF FIGURES	vii
LIST OF TABLES	viii
1 INTRODUCTION	1
1.1 Introduction	1
1.2 Weaknesses of the Present System	2
1.3 Objectives	2
1.4 Scope	3
1.5 Research Areas	3
1.6 Limitations	3
1.7 Organization of the Dissertation	3
2 BACKGROUND INFORMATION AND LITERATURE REVIEW	5
2.1 Background Information	5
2.1.1 Present System and Regulations	5
2.1.2 Financial Audit Analysis	5
2.1.3 Related Projects	6
2.2 Cryptography	7
2.2.1 Cryptographic Algorithms	8
2.2.2 Secret Key Cryptography (SKC)	8
2.2.3 Hashing and Blockchain	10
2.2.4 Digital Signatures	10
2.2.5 Multisignature	11
2.2.6 Merkel Tree	12
2.3 Turing Completeness	13
2.4 Decentralization	13
2.5 Peer-To-Peer Network	14
2.6 Consensus Algorithms	14
2.7 Bitcoin	16
2.8 Blockchain	17
2.8.1 Blockchain versions	18
2.8.2 Blockchain variants	18
2.8.3 Benefits of a blockchain	19
2.8.4 Transactions	20

2.8.5	Mining- Processing	21
2.8.6	Blockchain Usage	21
2.9	Ethereum	21
2.9.1	The Genesis Block	23
2.9.2	Ether	23
2.9.3	Gas Price	23
2.9.4	Solidity	23
2.9.5	Remix	25
2.9.6	Ethereum smart contracts	26
2.9.7	DApps	26
3	SYSTEM DESIGN AND METHODOLOGY	28
3.1	Methodology	28
3.2	System Context	29
3.3	System Architecture	31
3.4	Tools Used	32
4	IMPLEMENTATION	33
4.1	Frontend	33
4.2	Backend	34
4.2.1	Ethereum Private Network	34
4.2.2	Smart Contract Application	38
4.2.3	Smart Contract Server	45
5	RESULTS AND CONCLUSION	49
5.1	Discussion	61
5.2	Future Work	61
	REFERENCES	61
A	De-Centralized Secure Transparent Systems	65

List of Figures

2.1	Simple Encryption and Decryption Process	8
2.2	Secret Key (symmetric) cryptography uses single key [1]	9
2.3	Public Key (asymmetric) cryptography uses two keys [1]	9
2.4	Secret Key (symmetric) cryptography uses single key [1]	10
2.5	A Simple Hash Chain [2]	10
2.6	Sign and Verifying the Message[3]	11
2.7	How a Digital Signature is Created [4]	12
2.8	Merkle Tree for Efficient and Secure Data Verification	13
2.9	Centralized, Decentralized and Distributed Ledgers[5]	14
2.10	Different Types of Consensus Algorithms [6]	15
2.11	Top 5 cryptocurrencies based on market capitalizations [7] (as of August 2019) . . .	17
2.12	Blocks of Chain and Hashes	18
2.13	transaction validation process	20
2.14	ethereum blockchain Work progress	22
3.1	Work Flow Diagram	28
3.2	Context Diagram: Decentralized Secure Transparent prototype System	30
3.3	The architecture of the Decentralized Secure Transparent (prototype) System . . .	31
4.1	The architecture of the Decentralized Secure Transparent (prototype) System refer- ence from page no 31	33
4.2	Ganache Personal Blockchain for Ethereum Development	35
4.3	Install Metamask and Integrate with Ganache GUI	36
4.4	Mnemonic Seed Words on ganache GUI	36
4.5	General Settings on Ganache and Metamask!	37
4.6	Dapp portfolio	45
4.7	Remix - Solidity IDE	46
4.8	System Deployment	46
4.9	Decentralized Secure Transparent (Prototype) System	47
4.10	Metamask informations for deployment	47
4.11	Ganache informations for deployment	48
5.1	Sign the request and check new Administrator	49
5.2	Check new Administrator of the system	50
5.3	Sign and accept the Request by present Admin	50
5.4	Add Project Stakeholders by Administrator	51
5.5	Add Project Stakeholders by Administrator	51

5.6	Create Transactions by Any Stakeholder	52
5.7	Transaction Creator sign transaction by his privet key	53
5.8	Transaction created for transfer 140 ethers.	54
5.9	Sign the Transaction for transfer 140 ethers.	54
5.10	Created Transaction details	55
5.11	De-Authenticate the Transaction	55
5.12	De-Authenticated transaction	55
5.13	Transaction De-authenticate logs.	56
5.14	Transaction created for transfer 160 ethers...	57
5.15	Transaction creator approved for transfer 160 ethers...	57
5.16	Transaction approved for transfer 160 ethers...	58
5.17	Transaction approved for transfer 160 ethers...	58
5.18	Status of the Transactions...	59
5.19	Add more Authenticators for farther Authentications	59
5.20	60
5.21	Status of the Transactions...	60

List of Tables

2.1	Use Cases of Block-chain different sectors	21
2.2	standard metric for Ether Denominations	23

Chapter 1

INTRODUCTION

1.1 Introduction

Sri Lanka has changed a lot due to launched long and short term strategic and structural development plans. Which will help to make a transform to an upper middle income country[8]. It has high growth due to the strong contribution of the private sector in both investment and consumption as well as the government contribution to large infrastructure projects that will help to change the state of the country. Both Sri Lankan government and foreign countries are contributing to these projects as well as public and private organizations are also conducting a number of projects that attempt to improve the living conditions of the people by doing these infrastructure projects[8];

These projects also hold financial, economic, technical, institutional, social, political risks[9]. So we need to come up with new mechanisms to address and mitigate these external risks. In general, most projects seems less attractive or unsuccessful in generating expectations with respect to investment and weight of financial resources. As a result Sri Lankas economy finances and the society as a whole suffer from in wastage of resources[9]. There for. The real reasons for failure of these development projects are not known and debatable. Nowadays Sri Lankan government involve into many structural development projects[8]. Many of those projects are dealing with poor financial transactional methods and that will result a major risk on the project[9]. A buyer and seller when exchanging an asset for payment come to an agreement which is generally known as a financial transaction. The buyer and seller are separate parties that exchange goods for money. Financial audit is a review of financial transaction records [10] of a company which is later published as a report. The report includes what the reviewers saw during the audit and which areas regulators need to pay attention to[11].

According to researchers many projects have problems with cash flows because of poor financial planning and assessment[9]. Other risk factor include political influences that are generally not considers during the planning stage. According to the Transparency International [12], Sri Lanka currently holds 91th place out of 175 countries for the least corrupt countries. That suggests a huge amount of cash transactions take place in between as un-auditable events. Generally corruptions take a form of facilitation payments, bribe solicitation by government officers and politicians, nepotism and also cronyism. There an extremely high level of corruption carried out by those who are in charge of public procurement [12]. All these corruptions are finally added to the project cost and

that makes the project cost to be too high and might not deliver the expected outcome. Anti-corruption laws of Sri Lanka include Penal code[13] and bribery act[14] which criminalize corruption as well as attempted corruption in the form of extortion and bribery. Although there is no clear definition that differentiates bribery from facilitation payments under the prevention of corruption act[15]. But there is a high chance of escaping from these processes, due to the doubt in the integrity of the financial audit reports of such projects. Even Though the payment process of the government is complex and sophisticated many exploit the weakness in the system to scam money from these transactions. Law enforcement is restricted by a lack of resources and technical expertise, and the influence of politics[12]. In researchers point of view if they're a mechanism to immediate knowledge of the transaction and fraud can be immediately known and it can be mitigation.

1.2 Weaknesses of the Present System

- A transaction involves only a few parties and the transactions can be kept secret from other parties without transparency.
- Project financial management by one party, partitioning for approval and analysis of needs On the other hand, the transactions were carried out without transparency.
- In the event of transactions, all parties have no immediate knowledge of the transaction and fraud cannot be immediately known.
- Internal or external auditing takes place after the time of the transaction.
- In the long run, the audit report may be misplaced.
- Once the audit reports have been altered, other parties cannot be aware of it later.
- The payment process of the government is complex and sophisticated [16] [17] and which have much weakness in the system corrupt officials gain to exploit an above weakness for scam money from these transactions.

Therefore, this research is looking at the present time consuming manual Financial System to proposing a fast and trusted mechanism which can be mitigate above mentioned weaknesses by replacing a digital solution with smart contract built on block Chain technology [18] without compromising the prevailing legal framework of Financial Regulations[16] and Procurement Guidelines [17] by Sri Lankan Government.

1.3 Objectives

Basic Objectives of this research is minimizes mismanagement of existing manual process and proposed an efficient mechanism such as;

- Minimizes mismanagement of the existing inefficient, unproductive and untrusted manual system.
- Protect the Integrity of financial transaction data. Creating an efficient, auditable, transparent reliable transaction mechanism by cost effective manner.
- Create a digital signature for stakeholders for signing.

- Real time signature verification of transaction.
- Build conceptual design on blockchain smart contract to the fanatical transaction.
- Non repudiation of the transaction events.
- Protect legal framework of Financial Regulations by Sri Lankan Government.

1.4 Scope

Basic intention of this research is to identify and audit the financial flow of a certain project involved in many stakeholders. Proposed system can store encrypted version of transactional events occurred on project timeline and make that information available to all intended stakeholders for their reference.

1.5 Research Areas

In this system, Researcher uses an Ethereum (blockchain) [19] decentralized platform for building a “De-Centralized Secure Transparent Systems to Manage Financial Transactions”, Ethereum can developing decentralized applications in addition to its secure and decentralized technology. Which is tamper free, immutable and only a legitimate person with the correct person and signature can describe the blocks collect the relevant information within it. This research also provide to protect legal framework of Financial Regulations [16] and Procurement Guidelines [17] published by Sri Lankan Government.

1.6 Limitations

Prototype model of “De-centralized secure transparent Systems to Manage Financial Transactions” built on private Ethereum platform.

Present regulations for Financial Regulations [16] and Procurement Guidelines [17] published by Sri Lankan Government is complex and sophisticated, and which is difficult to implement the each and every requirements for digitalized system. Most of the cases government projects are done by contract base for their relevant requirement and this research is not include for any contract procedures. Researcher will follow only protect a ledger information’s and basic Financial Transactions process.

1.7 Organization of the Dissertation

The remaining chapters of this thesis are structured as follows:

Chapter two : Focused on the literature review, characteristics of Block Chain.

Chapter three : Discusses on building the system.

Chapter four : Implementation of the system.

Chapter five : Includes the final conclusion of the thesis.

Chapter 2

BACKGROUND INFORMATION AND LITERATURE REVIEW

This section will explore the technology behind Bitcoin and the blockchain, followed by an in depth look at smart contracts. Researcher expect to describe the background research on “De-Centralized Secure Transparent Systems to Manage Financial Transactions” and how the prototype system will aim to implement with core functionalities.

2.1 Background Information

2.1.1 Present System and Regulations

The Financial Regulations [16] and Procurement Guidelines [17] of the Government Sri Lanka is a subject that has been assigned to the Minister of Finance by the President exercising the powers vested in him under Article 44(1) a of the Constitution of the Democratic Socialist Republic of Sri Lanka [16]. In present Sri Lankan governments Ministries, Departments, Statutory Bodies, Government Corporations and all State Employees must follow the Financial Regulations (1966 revised on 1992) [16] when they processed the Financial transaction in an orderly manner. But the Financial Regulations will apply unless they have duly adopted their own comprehensive Financial Rules and Procedures. Furthermore that procurement of supplies, tender procedures and the execution of works can be neglected by the Head of Department which may, in appropriate circumstances but he should immediately be reported to the Secretary to the Ministry the Director General Department of Public Finance.

2.1.2 Financial Audit Analysis

Article 154 of the Constitution of the Democratic Socialist Republic of Sri Lanka provides the order for the Auditor General Department [10] to audit public sector institutions, accounts of all Ministries, Departments, and Statutory Bodies, Public Corporations and business or other undertakings. The Auditor General’s Department should be directly reported to the Parliament for provides an independent review of the accountability and performance of the public sector institutions also Department aims to meet the needs and expectations of the Parliament and ensure the regulatory, propriety and compliance with all the statutory and other regulatory requirements and the economy,

efficiency and effectiveness of the financial and other operations[11]. The importance of the financial audit analysis to an organization is;

- To reduce the fraud and controls misappropriation of organization assets.
- To trace the transactions to supporting documents and authorizations. To carefully review the records transactions, management reports, and also check the different legal documents required for running a business.
- To help the company to increase the strong accounting procedures and identifying weaknesses.
- To ensure that financial health of the company.
- To trace existing financial issues and to suggest necessary changes to the current procedures for a healthy strong financial management system.
- To determine additional factors which support the protection of the company's financial assets.
- To ensure the organization had a good habit of QOS and follow proper standards also policy for financial transactions.

2.1.3 Related Projects

There are many approaches introduced to overcome this problems related to manage financial transactions in Sri Lanka but none of them are tampered free trusted solutions.

- **Pronto-Xi Financials [20]**

Which offers a seamless view of the all the business financial activities such as accounts handling, payroll systems, assets management and the General Ledger.

The General ledger of the pronto-XI [20] is strong and flexible Ledger solution which provides a team with real-time access to the system for accurate, financial reports which is needed to support strategic decision making. And the pronto -XI provides a role based permissions when users associates with the systems. Pronto-XI can assist to analyse, record and classify the financial transactions and the general ledger to fully understand the financial of activities.

- **QuickBooks Enterprise [5]**

QuickBooks Enterprise[5] has been specially designed for users with experience. It includes advanced modules such as employee management, payroll, sales, inventory management and reporting. But entry level accountants may find it difficult to track fixed assets, direct credit card transactions, create custom and combined reports.

But which contain a general central database which is vulnerable to tampering, deleting records also access by role based permissions levels even no database encryption mechanism.

There are few existing systems which are similar to blockchain domain by comparing with the global industry but those systems are too expensive and it is difficult to manage the functionalities, but there is no any financial transaction management system connected to it;

- **Bitcoin [21]**

Bitcoin [21] is a purely crypto based currency, a form of electronic cash. It does not have a central authority (decentralized) which means that it can be sent from one user to another using the peer to peer network without any need for intermediaries. These transactions are verified using cryptographic mechanism by the network nodes and recorded in the blockchain which is a public distributed ledger.

- **Ripple's XRP Global Financial System[22]**

XRP system[22] was designed for banks. It is great for cross-border transaction. It uses a consensus protocol which records and validates every transaction without any mediator.

- **The Block-chain Model of Cryptography & Privacy Preserving Smart-Contracts**

Smart contracts[23] lets two parties that does not trust each other to transact safely without the help of a trusted third party. This transaction takes place over the block chain using decentralized currencies. If the contract is breached the blockchain ensures the honest party receives the required compensation.

2.2 Cryptography

Cryptography originated as the art of secret writing. It began thousands of years ago. The first use of cryptography was documented in 1900 B.C [1]. from the tomb of an old kingdom of Egypt. According to some experts cryptography was created soon after writing was invented and used for communication during times of war. A New Generation of cryptography emerged with the widespread use of data, computing and telecommunications. In today's business world cryptography provides the security necessary for communicating through a medium such as the internet that everyone has access to. Cryptography provides five primary functions[1];

1. **Privacy/confidentiality:**

Ensure no other parties read the message other than the intended receiver.

2. **Integrity:**

This assures the legitimate receiver that the received message has not been altered or modified.

3. **Authentication:**

Ensure that the sender's identity is identifiable.

4. **Non-repudiation:**

A process prove that either party cannot deny sending or receiving the data, message or information [1].

5. **Key exchange:**

The process of sharing the cryptographic keys between sender and receiver is called the key exchange [1].

Basic terms of cryptography;

- **Plaintext (P) :** This is the original message that can be understood by humans. Which is the message before the encryption process and after the decryption process [1].

- **Cipher text (C):** The message after the encryption and before the decryption. The data/message in a form that cannot easily be understood by humans.
- **Cryptographic Key (K):** An array of bits used by a crypto-algorithm to convert plain text into cipher text or vice versa [1].
- **Encryption (E):** The mathematical process of altering plain text data (plaintext data) into the something meaningless (ciphertext) [1].
- **Decryption (D):** the mathematical process of altering the cipher text message back to plaintext which is easily understood by human beings [1].

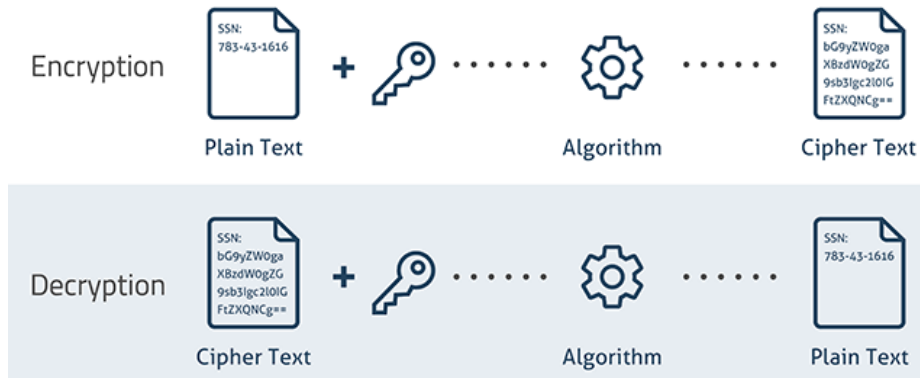


Figure 2.1: Simple Encryption and Decryption Process

Finally, Cryptography can be used in almost everywhere for applications such as banking, transactions, passwords, cards, and e-commerce transactions, etc. Cryptography is most closely associated with mathematical algorithms which is used to encrypt and decrypt messages, whereas cryptanalysis is used for investigating and breaching cryptographic security systems and gaining access to the encrypted cyphered messages [1]. Cryptology is the term referring to the broader study of both cryptography and cryptanalysis.

2.2.1 Cryptographic Algorithms

There are many ways of categorizing cryptographic algorithms. If it is classified using the number of keys used for encryption and decryption algorithms it is much more effective [1]. The three types of algorithms that need to be discussed which are

2.2.2 Secret Key Cryptography (SKC)

Here a single (same) key is used for both encryption and decryption. It is also referred as symmetric encryption. This type of cryptography provides privacy and confidentiality primary function [1]s.



Figure 2.2: Secret Key (symmetric) cryptography uses single key [1]

The sender uses a key to encrypt his message. Then he has to send both the message and the key to the receiver so that he can use the same key to decrypt the message he received. Here it is important to send the key using a secure way so that only the receiver will have access to this key. The weakness in this method is that if a third party knows the key they can also access the message.

2.2.2.1 Public Key Cryptography (PKC)

This algorithm uses a pair of keys. One key is known as the public key and it can be accessed by anyone. The other key is known as the private key, and is only accessible by the owner. This is called asymmetric encryption [1]. Encryption and decryption can be done by both keys if one key use to encrypt other must need to decrypt the message. This type of cryptography provides non-repudiation, authentication, and key exchange.



Figure 2.3: Public Key (asymmetric) cryptography uses two keys [1]

The sender encrypts his message using the receiver's public key. Therefore the receiver has to use his private key to decrypt the message. Nonrepudiation can be achieved by using the sender's private key to encrypt. Then the receiver has to use the sender's public key to decrypt the message.

2.2.2.2 Hash Functions

By using a mathematical transformation data can be encrypted in a way that is not reversible. They use no key which providing a digital fingerprint of the message [1].

Message digests and one-way encryption are synonyms for hash functions. Hash takes clear Text in conjunction with a Mathematical Algorithm and generate a fixed length bit string. It is not possible to recover both the content or its length. Hash algorithms are;

Typically used as a digital fingerprint of message or a contents of a file. By using this digital figure print we can find out if the file or message was altered by an intruder or virus. Receiver of the message and Hash run Message through Algorithm and compare results to Hash. If same message can be taken as authentic and unaltered. So hash functions are used by many operating systems to encrypt passwords[1].



Figure 2.4: Secret Key (symmetric) cryptography uses single key [1]

2.2.3 Hashing and Blockchain

It is important to know how blockchain Hashing works In order for new blocks to be accepted to an extent blockchain, a proof of work have to be generated. The proof of work is composed of letters and numbers fixed according to the desired outcome. This is expressed by the double SHA-256 hashing algorithm [2].

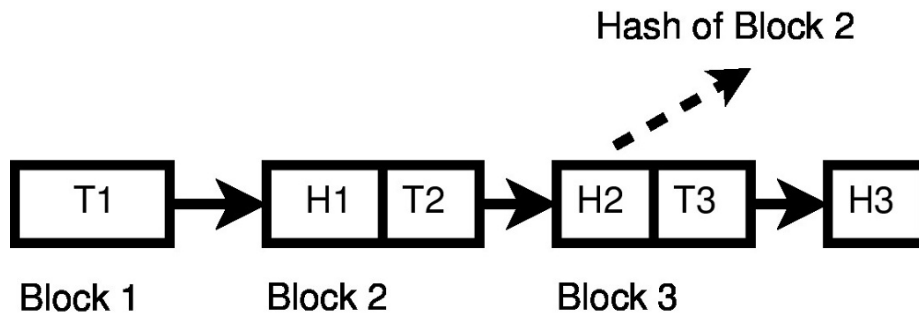


Figure 2.5: A Simple Hash Chain [2]

That means that once the target hash has been obtained, then the block is accepted into the public ledger by the consensus of other participating networks.

Because the blockchain only contains those transactions that have been validated, this prevents fraudulent transactions being added to the chain, or the problem of double-spending by reusing the same transaction twice [2].

Here the letter T stands for transaction. This diagram demonstrates the power of the chaining. H3 is generated by hashing the entire block 3, which contain the hash value of block no 2, which was generated using the hash of block 1 etc. considering that the hash function is one-way, it is impossible to calculate this chain from right to left (only calculated from left to right). There for any attempt to alter the chain parameters will be detected making the chain immutable. This is the useful property what we want.

2.2.4 Digital Signatures

A digital signature is a generated through a set of complex mathematical cryptographic calculations [3]. These Mathematical technique presenting the nonrepudiation, authenticity and integrity

of digital messages or documents. The digital equivalent of a handwritten signature or, a digital signature offers far more inherent security, and it is intended to solve the problem of tampering and impersonation in digital communications. Digital signatures are constructed using asymmetric key cryptography. Using an asymmetric key algorithm, such as RSA, two keys that are mathematically linked is generated one functions as the private key while the other functions as the public key [4].

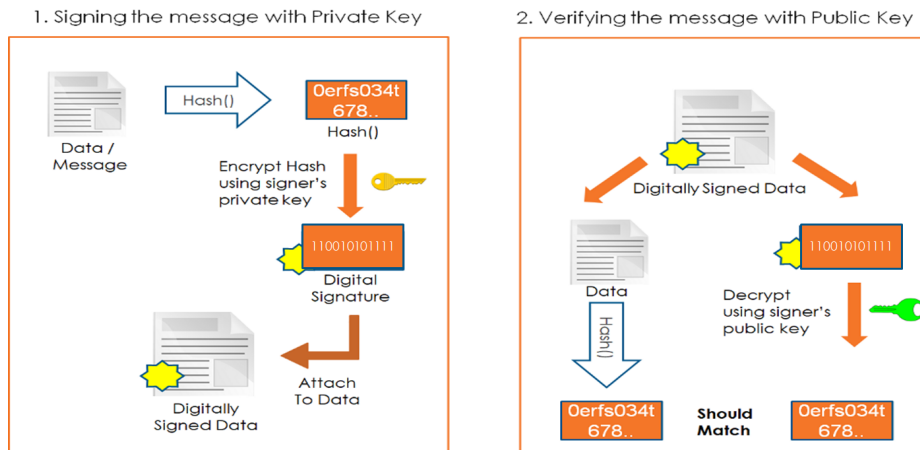


Figure 2.6: Sign and Verifying the Message[3]

A valid digital signature has three main functions. The first is the assurance that the message was sent by the claimed sender (proof of the sender's identity). This is known as authentication. The second is that the sender cannot deny sending the message. This is called non-repudiation. The third is that the message has not been altered during transmission. This property is called integrity. Today Digital signatures have become a common element of every cryptographic protocol that requires the above three functions. These include financial transactions, block chain applications, contract management systems, and everywhere else in which forgery and tampering needs to be detected [2].

To summarize hashing and digital signatures makes the blockchain unique. Because of hashing everyone can trust the blockchain state has not been altered. In the same way digital signatures ensure that all transactions are made by the rightful owners. This is what guarantees the blockchain is not compromised or corrupted. So Digital signatures are a key component of the blockchain for securing data and auditing the transactions.

2.2.5 Multisignature

Multisignature [4] is a digital signatures scheme for more than one signature is needed to approve a transaction. Multisignature (Multisigs) are used by many cryptocurrencies such as Bitcoin and Lisk as well, concept of the Multisignature to provide dividing the ability to make decisions group of peoples or parties which provide better security for the transaction. This feature provide transactions makes the system significantly safer and secure, both by hackers and those who somehow gained access also prevent fraudulent transaction of smart contract .

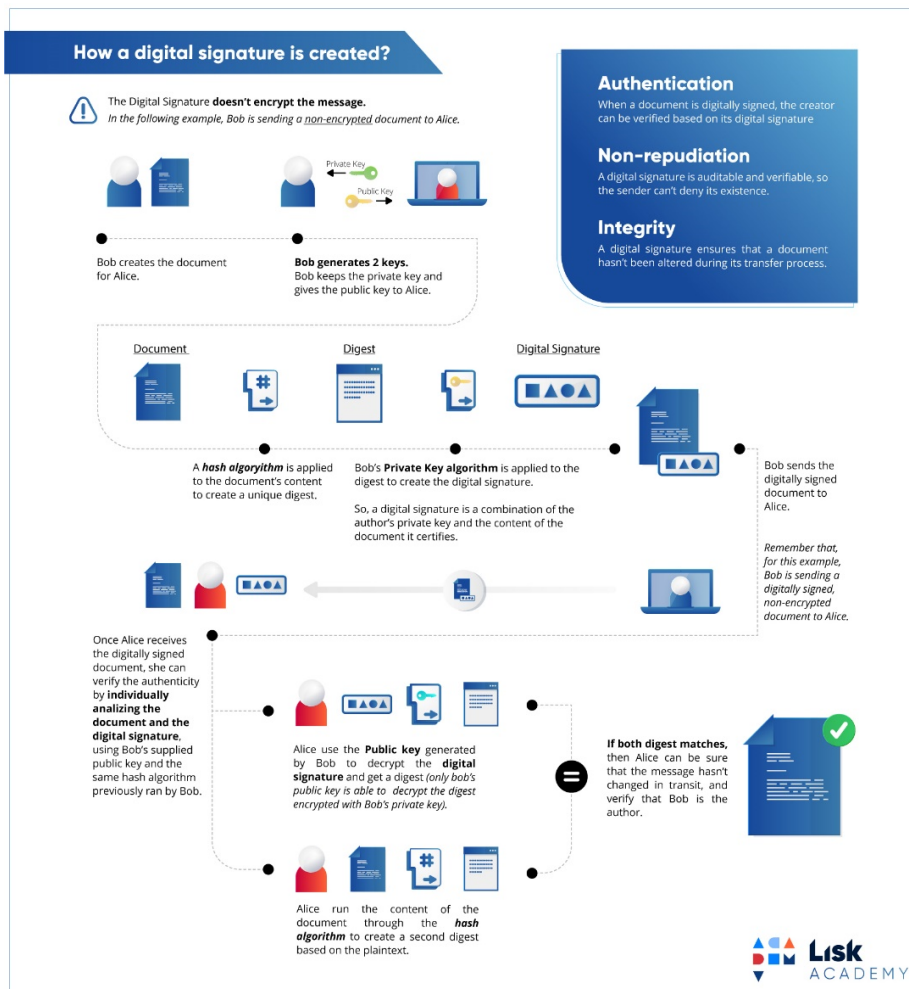


Figure 2.7: How a Digital Signature is Created [4]

2.2.6 Merkle Tree

The idea behind a Merkle tree [24] evolve from the concept of hash list. It soon became a fundamental part of blockchain technology. It has a tree like structure where each parent node is generated by hashing two child (leaf) nodes. Child nodes (leaf-node) are generated by hashing block chain of data. In this tree each parent node has only two-child nodes making the branching factor equals to two.

To assure effective and secure data-verification of content in a large body of data such as in distributed systems merkle trees are generally used. Hashes encode files in to much smaller content than the original file. This is what makes this system Efficient. Entire data from a file is replaced by a much smaller hash of fixed length. P2P networks such as Tor, Ethereum and Bitcoin uses this function.

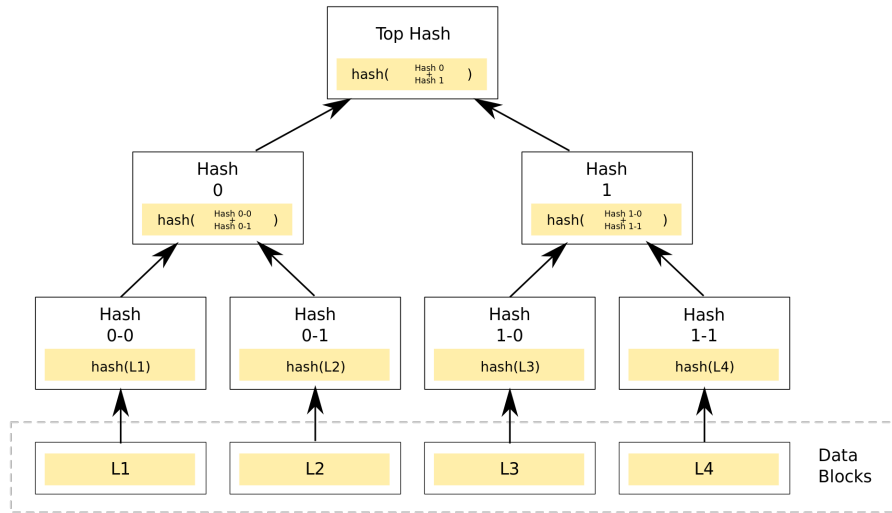


Figure 2.8: Merkle Tree for Efficient and Secure Data Verification

L1 though L4 are blocks of data which are then hashed to create leaf hash nodes. Then these nodes are paired and hashed recursively until root node is reached. Therefore the hash of the root node is calculated using all other nodes in the merkle tree that's why it is called the merkle root [24].

2.3 Turing Completeness

The concept behind the modern computer was proposed by a mathematician called Alan Turing in the 1930s. This hypothetical machine was called the Turing machine. In theory it was able to perform any mathematical or logical calculation as it had unlimited memory. Ethereum is one such network that has the potential to mimic Turing Completeness [25].

The term turing completeness refers to a system or device that can calculate anything if memory is available.

2.4 Decentralization

Since it doesn't rely on a central point of control the blockchain technology is said to be decentralized [5]. It means each node is running autonomously in the system. The main difference with a distributed system is that in a distributed system there are dedicated set of nodes to control other nodes. But in a decentralized system no one takes commands from each other. No concept of client server or master slave. Everyone has equal decision power.

Since there is no central authority the system is fairer and considerably more secure. Instead of a central authority, blockchain uses an advanced consensus protocols[26] throughout the network, to

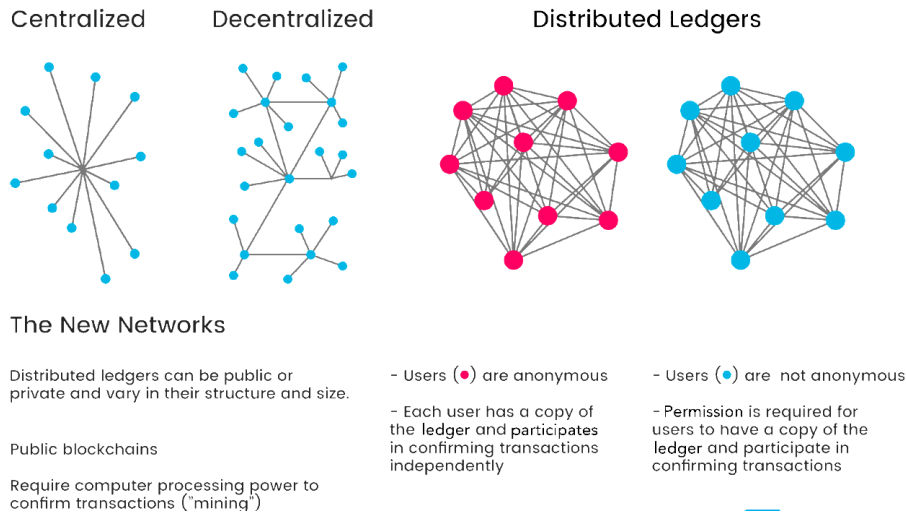


Figure 2.9: Centralized, Decentralized and Distributed Ledgers[5]

validate and record transaction data in an immutable fashion. As a blockchain is equivalent to a digital ledger it is very important that the information is accurate.

2.5 Peer-To-Peer Network

A Peer to Peer network[5] is a very important part of how blockchain technology. Blockchain technology has a system of recording and confirming transactions using consensus protocols. Here, each node in the peer to peer network holds a complete record of transactions. As a result, a central recording system does not exist, instead each participant has a record of all transactions ever made. This is the same system used by Bitcoin.

2.6 Consensus Algorithms

Decentralization and security are very important features of Blockchain. Consensus algorithm [6] is what helps a network with multiple nodes achieve reliability. Finding a solution to the consensus problem is essential in multi agent systems and distributed computing to ensure that the next block in the blockchain is fully validated and secured. There are different kinds of consensus algorithms in existence, each with different fundamental processes [6].

The consensus algorithm should do two things, Make sure the next block added to the blockchain is the only version and inhibit anyone from disrupting the system and forking the chain. There are many consensus algorithms available some of them are;

Proof of Authority Algorithm (PoA)

Only some nodes with a particular key holds the status of validators and only validators are allowed to add new blocks [6].

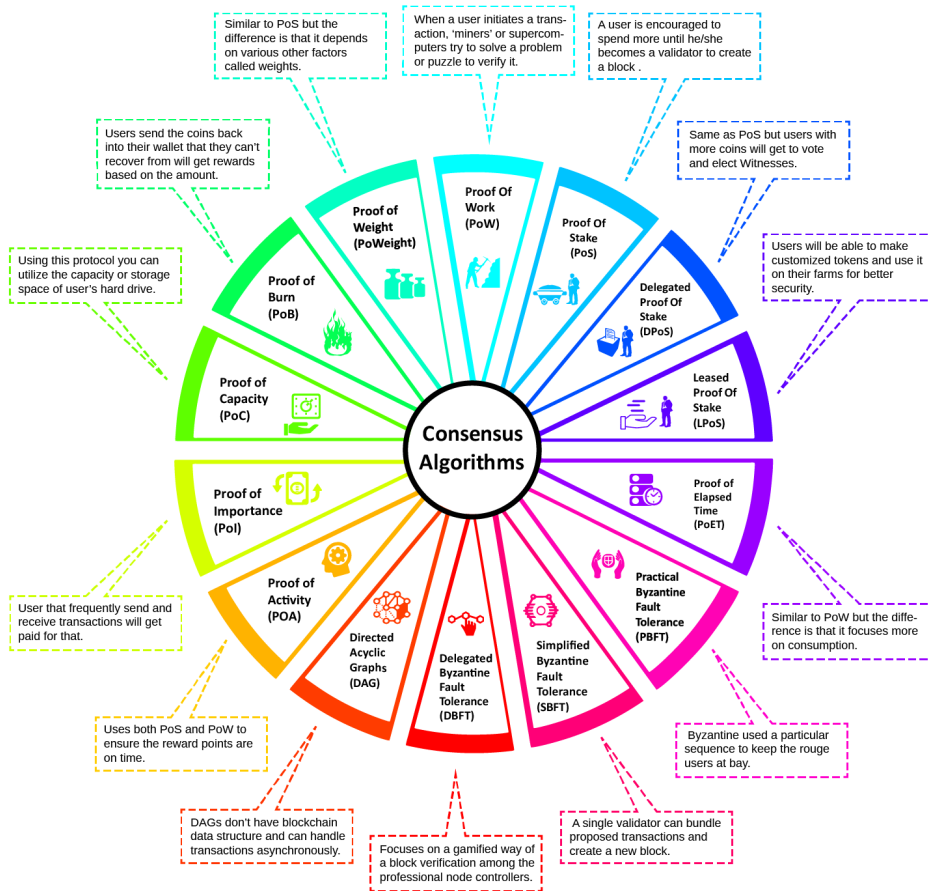


Figure 2.10: Different Types of Consensus Algorithms [6]

The Proof of Work Algorithm (PoW)

The nodes in this system solves Cryptographic complex computation to verify each block before it is added to the blockchain. This calculation can take around 10 minutes [6].

The Proof of Stake (PoS) Algorithm

This algorithm has validators who are selected based on criteria such as coin age and economic stake. It is important to note that this algorithm requires much less energy due to its selection criteria [6].

The Byzantine Fault Tolerance (BFT) Algorithm

This algorithm defends against a catastrophic system failure by reducing the influence that malicious nodes have. Malicious nodes are failing nodes or nodes that propagate incorrect information to other Peers [6].

Proof of Burn (PoB) Algorithm

This algorithm encourages miners to prove their long-term commitment by burning their coins. Burning happens when a coin is sent to an unsend able address. Only miners who have proved their long-term commitment to the coin will receive incentives much later on [6].

2.7 Bitcoin

The first decentralized cryptocurrency bitcoin and blockchain technology was introduced from the original white paper "Bitcoin: A Peer-to-Peer Electronic Cash System" in October 31st of 2008 by, pseudonymous developer called Satoshi Nakamoto [21]and his team was created it in 2009. It used SHA-256, a cryptographic hash function and an electronic payment system utilizing cryptographic calculations. The idea behind it was to create a transaction method, not depending on of any central authority or single administrator that has the ability to be exchanged electronically in a secure, verifiable and immutable way [27].

Bitcoin to Bitcoin transactions are made through a peer-to-peer (P2P) network by digitally exchanging anonymous hash codes that are heavily encrypted. Transactions are verified and monitored by the P2P network. Bitcoins are stored in a digital wallet belonging to each user. Digital wallet is a program for storing currency. It has an address which is also unknown as the public key of the wallet. The owner can authorize this transactions using his private key which is only known by him. Wallet also contains previous transaction details.

By design the bitcoin network can generate up to 21 million bitcoins. The network is capable of dealing with inflation and regulating itself. Spending bitcoin involves sending bitcoins from the customer's wallet to the vender's wallet. As of now the bitcoin is valued at \$ 10139.80 [7] but like in stocks the value fluctuates quickly. Features of Bitcoin are listed below [27].

1. Direct payments from P2P through the network
2. Third parties are eliminated and trust is replaced by verification.
3. Irreversible Transactions. This protects sellers from deception. Buyers can be protected by escrow mechanisms.
4. Time map servers on the P2P network provides proof on the chronological order of transactions. This is a secure system given that honest participants as a group have more computing power than the attackers.

Since external third parties have the power to commit fraud Nakamoto argues that the verification process is better than having trusted third parties when it comes to financial transactions [27]. The fact that transactions cannot be reversed builds trust and confidence in the system. It also minimizes the ability to commit fraudulent acts through the system. Since many nodes have to confirm a transaction before it is carried out the users feel confident in the validity of the irreversible transaction adding to the overall confidence in the system.











#	Name	Market Cap	Price	Volume (24h)	Circulating Supply	Change (24h)	Price Graph (7d)
1	 Bitcoin	\$185,476,376,143	\$10,363.23	\$18,165,410,374	17,897,550 BTC	2.58%	
2	 Ethereum	\$20,512,408,085	\$190.85	\$6,978,623,158	107,476,600 ETH	0.65%	
3	 XRP	\$11,687,367,559	\$0.272372	\$1,153,895,219	42,909,539,227 XRP *	0.31%	
4	 Bitcoin Cash	\$5,605,688,191	\$311.98	\$1,365,163,125	17,968,075 BCH	1.66%	
5	 Litecoin	\$4,699,276,393	\$74.46	\$2,839,672,934	63,115,362 LTC	1.97%	

Figure 2.11: Top 5 cryptocurrencies based on market capitalizations [7] (as of August 2019)

Up to now double spending has been a problem in the digital world as a digital asset can be easily copied and re-used [27]. Bitcoin takes care of this problem by the creatively utilizing cryptography and economic incentives. When fiat currencies are transferred through a public network in the traditional system it is the banks that help mitigate the double spending problem. This gives banks power and control over the system. Since the distributed open network is not owned by anyone it can be considered as a system with much more integrity.

The blockchain to be used is Bitcoin, there is at first no difference between Bitcoin and blockchain. Many used cases based on blockchain are in the financial sector, however a Bitcoin-like cryptocurrency Ethereum has added functionality for smart contracts[19].

2.8 Blockchain

As suggested by the name, blockchain contain information in the form of a chain of blocks. It was created for cryptocurrencies [26]. It functions as a public distributed ledger that transactions are recorded in an immutable fashion. Even with the system continuously growing all records of blocks are synchronized with identical information. It can be seen as a decentralized ledger that tracks transaction effectively in a permanent and verifiable way. It is important not to confuse Bitcoin with the blockchain. Blockchain is the underlying Technology in bitcoin. Bitcoin is just one digital token. There are many others. The blockchain is used to track the integrity of these tokens. Simply, Bitcoin cannot exist without blockchain, however, blockchain can exist without Bitcoin [26].

Each block has a header that includes the hash of the former block's header generating a chain like structure between blocks. If any part of the block is altered, the value of the hash will also change, which will change the value of hash referenced in the next block as well as all other blocks that follow. This is what makes the blockchain immutable. The concept of proof of work is what stops an attacker from rewriting the whole chain.

The transactions in a block are pared and hashed. The resulting values are then again pared and hashed. This process is continued until only one hash is obtained. This single hash is called the

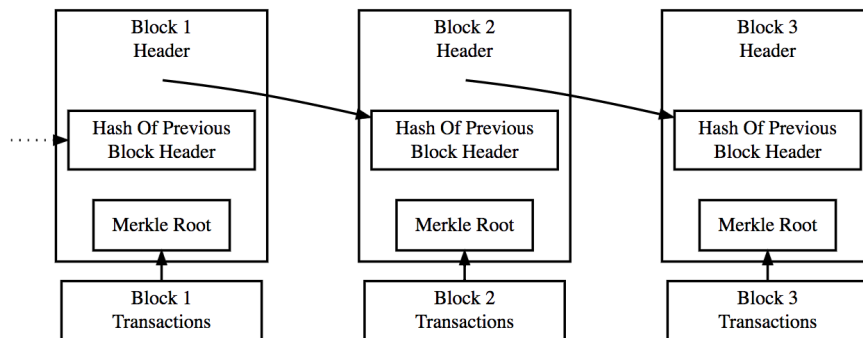


Figure 2.12: Blocks of Chain and Hashes

‘Merkle root’. The process as a whole produces a tree-like structure called the Merkle tree. This is why transactions cannot be modified since any modification will Result in a modification in every single hash value propagating up the tree [23].

In summary, blockchain is an Internet-based software protocol that acts as a digital ledger that is constantly updated and is run on a distributed network which can be accessed by anyone at any time. The blockchain is a single source of truth since it is impossible to hack the blockchain.

2.8.1 Blockchain versions

Generation 1.0 : Crypto-Currency

Cryptocurrencies are the most famous application of the (distributed ledger technology) DLT. Financial transactions can be performed on the blockchain using cryptocurrency. Bitcoin is the most popular currency for this generation of blockchain [26].

Generation 2.0: smart contract

A smart contract can be seen as small computer program data available in the blockchain. They can be used as a replacement for traditional contracts. The processes facilitation verification and enforcement is automated in smart contracts. When a predefined criteria is fulfilled the smart contract will automatically enforce the previously agreed transaction[26].

Generation 3.0: D-apps

D-apps stands for decentralized application. They have a back end code processing on this decentralized P2P network. They can have a front end code as well as user interfaces that can make a call to its back end like a traditional app[26].

2.8.2 Blockchain variants

Public Blockchain

Public blockchain ledger is available on the internet for everyone to see. Anyone can verify and add block transactions to the blockchain. Public networks have incentives for people who mine[26].

Private Blockchain

When a single organization is using blockchain it is called a private blockchain. Only pre-specified people are allowed to verify and add transactions to the blocks in the blockchain. However, anyone who access the network has the ability to see these blocks[26].

Consortium Blockchain

A group of organizations can use this kind of blockchain to perform cross organizational transactions. The ledger has the option to be Restricted or open to selected parties. The blockchain is controlled by pre-authorized known[26].

2.8.3 Benefits of a blockchain

Blockchain technology is rapidly being adopted all over the world especially in the IT industry thanks to its wide range of benefits some of these benefits are listed below[26].

Resilience

The architecture of the blockchain is often replicated, therefore, it can function even while a massive attack Takes place against this system.

Time reduction

Blockchain significantly reduces the Time taken to process transactions since it does not require a long time for settlement, Verification, and clearance since the ledger is available for all stakeholders to view.

Reliability

Since all processes searches certification verification and identification is done by the blockchain there will be no double records and transactions will be accelerated while reducing the rates.

Unchangeable transaction

The use of hashing in the blockchain make sure the transactions cannot be altered. All transactions will be entered in chronological order and once it is added to the blockchain it cannot be deleted or altered.

Fraud prevention

Since all information in the blockchain is accessible to everyone this system prevents possible losses due to fraud or embezzlement.

Security

A general database is vulnerable to be brought down by attacking a specific location. Thanks to DLT, the blockchain remains active even if many nodes fail since each party holds the original copy of the chain.

Transparency

Whatever changes that occur in the blockchain is publicly viewable by anyone, therefore, blockchain offers greater transparency.

Collaboration

Because of above-mentioned features such as reliability transparency and security different parties can interact with each other directly without a 3rd party for mediation.

Decentralized

All blocks in the blockchain follow the same standard rule when exchanging information. Therefore blockchain makes sure that all transactions are authorized. After that all valid transactions will be added to the next block in the blockchain.

2.8.4 Transactions

A transaction is the process of transferring some kind of value. In this case tokens (some kind of data) are been transferred between wallets. These transactions are recorded in the blockchain. Wallets have a secret code known as the private key or a seed. It is used to authorize transactions. This is what provides mathematical proof regarding the identity of the owner. Another use of this signature is that it prevents the record in the blockchain from being changed after it is entered. All transactions are usually confirmed within 10 to 20 minutes after the transaction is broadcasted [28].



Figure 2.13: transaction validation process

transaction validation process [26]

1. A Blockchain transaction is requested by bob. This transaction may include cryptocurrency, records, contracts or other information.
2. Next the request broadcasting takes place throughout the peer to peer network.
3. Nodes in the network validate the transaction using an algorithm.
4. After the transaction is validated and confirmed by multiple nodes in the network it is added to a block which is then added to the blockchain. This record is permanent and can never be changed. If the validation fails the transaction will not take place.

2.8.5 Mining- Processing

Mining can be seen as a reward system for those who run the nodes in the blockchain. This consensus system requires miners to follow a strict set of rules and guideline to be rewarded[28]. This ensures that transactions are processed chronologically and assures the systems neutrality. A block is only accepted in to the chain if the majority of the nodes agree. The strict set of rules also prevents old blocks from being modified because doing so will invalidate the blocks that follow. Mining creates competitive environment therefore no one can add new blocks consecutively. Because of this no one can control what is in the blockchain or replace any information.

A transaction fee is required for a blockchain (in the bitcoin) transaction. The miners will receive the taxes of all transactions in the block. For bitcoin the price of a block is reduced by half every 210000 blocks [28]. This takes four years roughly. It used to be 50 bitcoins but now it has been reduced to 12.5 bitcoins.

2.8.6 Blockchain Usage

The decentralized blockchain technology is now most trusted and secured ledger technology which can be applied in several domains for providing a trusted service for their clients which categorized as follows [26],

Domain or Industry	Service or Application
Marketplaces	Transaction management , Billing and procurement management
	stock management in the supply chain network
	online secure payment and booking systems
	Decentralized markets store
The government sector.	Registry, Notary and identity service
	Voting system
	Tax and payment systems
	Tender and contract management systems.
IOT sector	Document digitalization such as Properties and ownership management service
	agricultural and sensor networks,
	smart and sensor home networks,
Finance and accounting domain	Personalized robotics and consumer components.
	Digital payments system
	Decentralized markets store
	payments and remittance
Health and pharmaceuticals domain	Inter-divisional accounting
	Health information Bank
	Health and hospital management systems
Science and development	Digital health wallet Smart property
	p2p resources management
	super computing and data processing

Table 2.1: Use Cases of Block-chain different sectors

2.9 Ethereum

Ethereum is a second generation cryptocurrency [19] which was introduced in 2013. The main goal of Ethereum is to provide a platform for decentralized applications. These applications use the concept of smart contracts. Therefore the application is immune to third-party interference, fraud

or censorship. Since this is a blockchain Technology It has no possibility of downtime. Ethereum network is fueled by its underlying currency Ether. It has a fairly new programming language that is executed by an internal virtual machine (EVM) This language is developer friendly. With the help of the supporting community, It is becoming widely used and adapted. The custom built blockchain that runs these applications is very powerful and has infrastructure that is shared worldwide. It can move values and ownership of property. Therefore, developers can create markets, move funds according to previously agreed upon conditions, store Debt information in registries, etc. without a middleman, reducing the counterparty risk. The figure shows the schematic diagram 2.14 for ethereum blockchain.

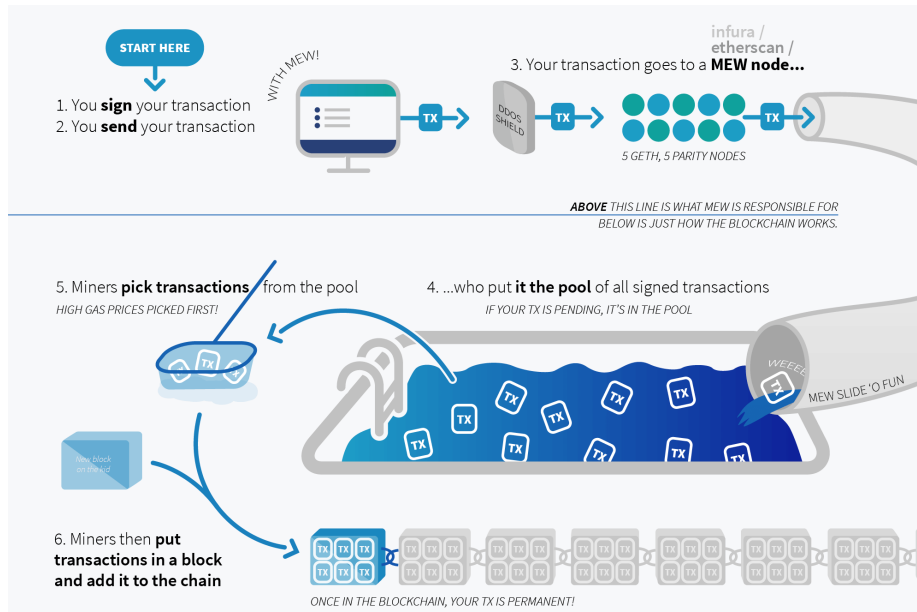


Figure 2.14: ethereum blockchain Work progress

The process of validation is what causes smart contract to run automatically. This process is executed by full nodes autonomously. Full node downloads and validates the whole blockchain without trusting other nodes. Portable devices, however, use lightweight clients that stores the blockchain partially and depend on full nodes for transaction validation [26].

The cryptocurrency used in the ethereum platform is Ether. It is used to store and transfer monetary value as well as to pay for computation and transaction costs. Gas is used to measure transaction costs as a way to avoid saturation due to meaningless junk transactions. The consensus algorithm for the ethereum network use proof of work protocol and consumes a lot of energy causing environmental concerns. To solve this issue the network was expected to move into proof of stake system which was expected to be fully implemented by 2018 [26].

2.9.1 The Genesis Block

Genesis block or Block 0 is the first block of a block-chain. A specific folder is created to save this block as a document in the .JSON format. This is where it will be called from.

2.9.2 Ether

The currency token used in the ethereum platform to pay transaction fees is called ether. in order to deploy contract code developers need to provide ether. When users invoke transactions on a contract they have to either burn or spend ether. Ether can be exchanged into other Fiat currencies as well as other cryptocurrencies in various cryptocurrency exchanges. Ether can be divided into smaller units; the smallest being wei [29]. It is 10^{18} th of an ether[26] and table 2.2 shows that the standard metrics of ether.

Unit	Wei Value	Wei
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000

Table 2.2: standard metric for Ether Denominations

2.9.3 Gas Price

The ethereum network has associated price in ether gas, which users need to pay to initiate a transaction. Gas is a unit of computation an internal pricing unit that separates the market price of ether from the cost of transactions. Gas is not a currency. It's the price for the simplest operation executed on the blockchain is represented by 1 unit of gas. Depending on the complexity of the operation smart contracts have different gas costs [26].

Gas price is the fee for utilizing a single unit of gas which is dynamically adjusted to the fluctuating values of ether. Insufficient Ether balance, as well as lack of gas, can cause the transaction to rollback.

The originator of the transaction can decide the gas price. miners can also decide whether they want to include the transaction in their blocks. The start gas value or gas limit determines how many computational Steps a transaction is allowed. Since more memory and storage is used to execute more lines of code in a contract more Complex contracts have a higher gas limit. Gas limits and transaction fees help prevent faulty code such as infinite loops from execution saving computational resources on the network. Gas limit discourages denial-of-service attacks on the network as well.

2.9.4 Solidity

Although ethereum supports many different languages for writing smart contract. Solidity [30] is regarded as the official language maintained by the ethereum project. This high-level language is object-oriented as well as contract oriented. Solidity system is the static type with a syntax very

similar to java script. Solidity main construct is a contract. It contains fields, functions and function modifiers as well as struct types, enum types, and inheritance.

Ethereum address is a interesting data type which is 20 byte long. It can represent an external account or a contract. This data type has members that are predefined which has the ability to transfer either or check the balance using a contract or a call function. Structs, enumerations and byte arrays which are able to store multiple data types are supported by Solidity. Also supports mapping which is seen as a hash table that could map any data type to another data type. Every declared variable produces a accessor function.

The most important concepts of a contract are described in the following subsections

2.9.4.1 Solidity Variables and Data Types

State variables store values in a contract permanently . The data type of the variable needs to be specified. Elementary data types used in solidarity are listed below. They can be combined to create more Complex data types [30].

- **Booleans**

boolean values (true, false) are stored using this data type.

- **Integers**

Different subtypes of signed (int) and unsigned(uint) integers with different byte lengths are stored using this data type.

- **Address**

Ethereum addresses of 20 bytes are stored using this data type.

- **Fixed-size byte arrays**

Arrays that varies from length 1 to 32 are stored using this data type.

- **Dynamically-sized arrays**

“bytes” type arrays, dynamically-sized byte arrays, string type arrays and, dynamically sized UTF8-encoded strings are stored using this data type.

- **Struct Type**

These are custom defined types that have the ability to group several variables

- **Enum Types**

These are utilized for creating finite set of values with custom data types.

2.9.4.2 Visibility for Functions

Solidity has functions and variables in the Global namespace which provides information on the status of the blockchain and executable unit codes in the contract. Solidity holds four types of visibility functions [30] which are.

- **External**

This function is a specification in a contract which can be called by other contracts. But the contract containing the function cannot call itself.

- **Public**

This function can be called by both the contract containing it as well as other contracts.

- **Internal**

Only the contract containing this function or its derived contracts can call it.

- **Private**

Private functions are only available for the use of the contract containing the function. Derivative contracts cannot use this.

2.9.4.3 Other Impotents Notes

Function Modifiers

Function modifiers [30] can change the behavior of a function and make minor changes to the semantics of a function. They make sure that a function can be executed only in a specific context.

Events

Events are interfaces that grant AVM logging facilities. Applications can subscribe to events and monitor the state of a contract without directly interacting with the contract itself.

2.9.5 Remix

Remix [31] is a web-based IDE. It writes solidity smart contracts, deploys and run them. It comes with a running environment, debugger and testing facilities. It is very secure since it analyses the solidity code to reduce coding mistakes and identify vulnerable coding patterns [30]. Examples of such vulnerabilities are transaction origin usage, block-hash usage, dependence on timestamp, reentrancy, and patterns with high has costs. Formal verification such as deductive program verification and theorem proving are used for remix Security Analysis.

Remix IDE can be access through the web browser without any kind of special installation by visiting <https://remix.ethereum.org/> [31]. You can access a complete IDE which have a code editor and various panels for compiling, debugging and running the smart contracts.

2.9.6 Ethereum smart contracts

Smart contract process inputs automatically and perform logical calculations based on predefined criteria to provide results. Obvious next step for this application is to improve the complexity of the agreements. The blockchain stores, executes and verifies the smart contract code. The second generation of blockchain facilitates trades using cryptocurrencies and smart contracts. Nick Szabo proposed Smart contracts in 1997. The main objective of this contract is to automatically execute the transaction once all terms and conditions are met. the simplest smart contract is shown in below [30];

```
pragma solidity >=0.4.22 <0.6.0;

contract Transparent_System {
    address public Admin;
    Transaction[] public Transactions;

    constructor () public payable{
        Admin = msg.sender;
    }
    modifier Only_Admin {
        require(msg.sender == Admin);
        _;
    }

    function Change_Admin(address New_Owner) Only_Admin public {
        Admin = New_Owner;
    }
}
```

As a result of transactions stand-alone contracts is saved in the blockchain itself for storing executable codes, data, or ether. Smart contracts have the ability to interact with other contracts and create new contracts if necessary. They have a low bar of entry for users due to low legal and transactional costs. Currency is used by the consumer himself, therefore, security can be an issue. if you send currency into a Buggy contract it can be lost forever. the smart contract [32] has the ability to hold assets which will only be released once certain conditions are met.

Contracts act on behalf of the users with the exception of initiating a transaction. This limitation was introduced to dodge DoS attacks against contracts on the blockchain. The user has to pay the gas required to execute the contract before the process begins. Since programs can refund users, the contracts that people can use for free is a real possibility.

2.9.7 DApps

The Dapps stand for decentralized applications. There are many apps available and also in development. The average user of the blockchain usually interacts with the ethereum network using these apps. Dapps can be viewed from the state of the Dapps web page. Marketplaces, exchanges and betting applications are some popular features of the Dapps (most Dapps are still development). To access these features the user must connect with the ethereum network. For this, the user must download an ethereum client which is usually available for download from the internet [33].

The go version (geth/go-ethereum) is the most common implementation while other versions such as versions written in Rust (parity), C++ (cpp-ethereum) are also available. The client downloads the blockchain and verifies blocks, manages ethereum accounts using command line interfaces, sends Ether and deploys smart contracts. Mining can also be performed by clients.

The end users usually use the Mist Browser which has go client bundled into it. Developers have the option of using browsers with the client through the JSON RPC API. Most apps use JavaScript API [33].

Chapter 3

SYSTEM DESIGN AND METHODOLOGY

This chapter will discuss the requirements and provides a comprehensive explanation on the methodology which followed by the researcher to addressing for developing a prototype of Decentralized Secure Transparent Systems to Manage Financial Transactions for the Sri Lankan government organizations.

The system is a blockchain-based system which is capable to Direct Payments for any stakeholders in the particular Project. The main goal of the system are introduced a proper transaction ledger and securing that ledger from bogus users for gain fraud transactions.

3.1 Methodology

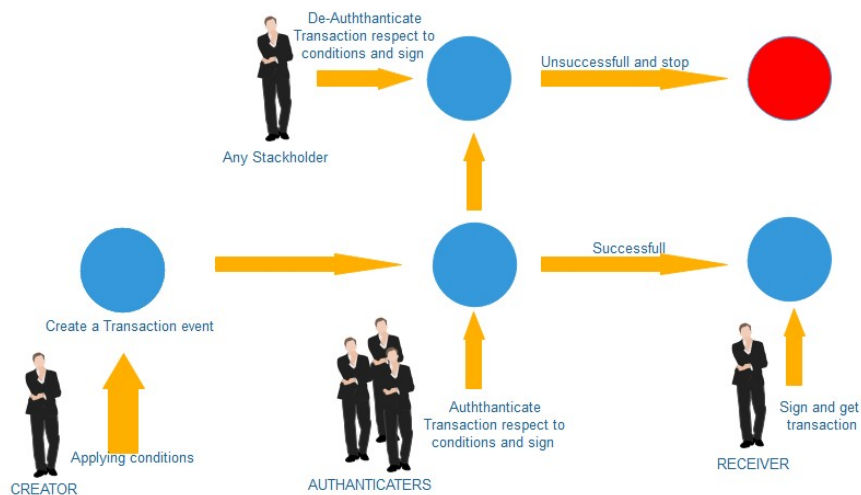


Figure 3.1: Work Flow Diagram

This prototype is based on the blockchain and related technologies. The basic intention of the system is to create, store and validate the legitimacy of financial transactions in a way that is tamper free, immutable and guaranteed the integrity. The prototype system which runs on decentralized private network, only a legitimate user with a valid cryptographic signature can create and validate the financial transactions. Also the receivers of the transaction needs to have valid cryptographic signature in order to accept and cash out from that transaction. This strong cryptographic mechanism prevents bogus users and miners from altering and creating contents of the ledger. So this system provides trusted, transparent and reliable mechanism which can be audited by any number of parties in order to verify the overall integrity of the created transactions.

In order to replace legacy systems like cash book and ledger adaptable with existing legal framework of financial regulations of Sri Lanka (Section 2.1.1). And aligned with financial auditing (Section 2.1.2) requirements, proposed system must have controls required to fulfill requirements of legal framework and auditing procedures. In order to fulfill above requirements, researcher was identified some of basic criteria considered when design the prototype system.

When the system is started by the Administrator, he became an owner of the system, Administrator should have to add all the stakeholders by using their pair of keys. Administrator hold the user manipulation rights Such as add or remove stakeholders also he can Transfer the Administration authority to another one but if he not become a stakeholder he cannot create transactions.

Transactions can be created by any stakeholder by application. The creator of the transaction should decide Authenticators also conditions to comply by Authenticators for genuinely successful transactions. The transaction data is sent to all stakeholders for further processing. Authenticators can authenticate the transaction with respect to the conditions and also add more conditions for further processing or can also any stakeholder can de-authenticate the transaction if it is not authenticated by his signature. If all Authenticators proof legitimacy of the transaction will be successful and value (if the creator set the value) should transfer recipient wallet by automatically or transaction can completed manually by proper banking check. Through this process internal audit team can verify each and every transaction by comparing relevant wallet hashes.

So this system provides trusted, transparent and reliable mechanism. Furthermore Integrity and authenticity of the transactional data can be preserved by introducing appropriate encryption and hashing mechanism. For each wallet data of stakeholders and third party or internal audit team or any number of parties can be verify each and every transaction by comparing relevant wallet hashes. Non reversible hashes and encrypted wallet information can further prove non repudiation of the transaction events with certain owner who responsible for the action.

3.2 System Context

The Decentralized Secure Transparent prototype Systems is act together with few number of actors also exterior software systems. This section is introduced by these actors and software systems. According to the context diagram as shown in figure : 3.2 which defines the Decentralized Secure Transparent prototype System with the inputs and outputs from its set of actors.

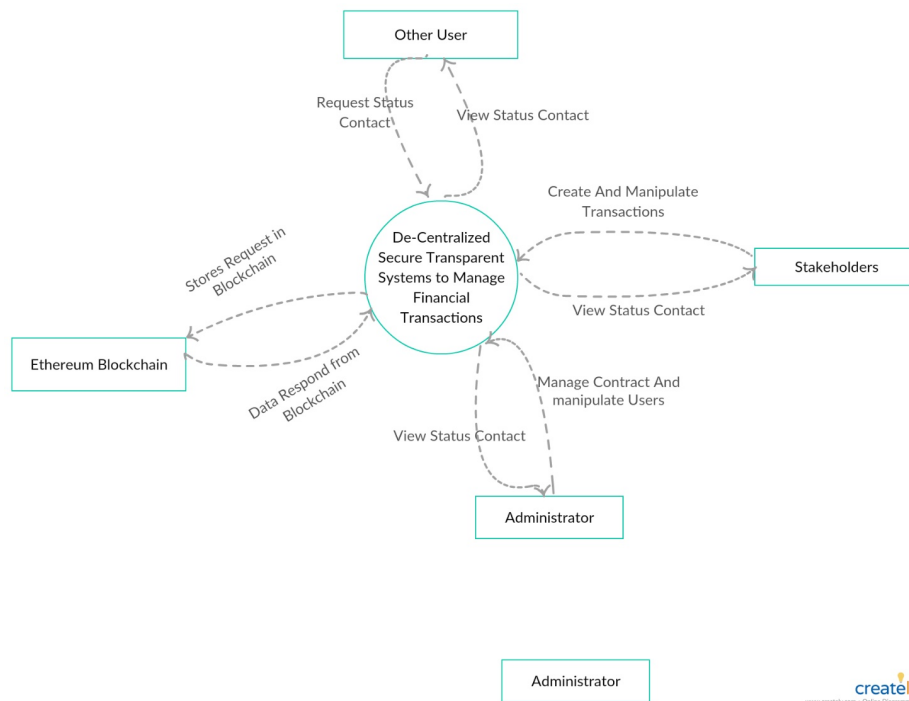


Figure 3.2: Context Diagram: Decentralized Secure Transparent prototype System

Administrator

Any one deployed the system he is become an administrator of the system as well as he is the owner of the system, afterwards he can Transfer Administration authority to another. Administrators primary job role is manipulate the stakeholders.

Stakeholders

Stakeholder is the main actor of the system and he has many user roles such as Authenticator, De-Authenticator. So basically stakeholders can create Transactional request and manipulate the Transaction. Once they have created a transactional request and the creator of the transaction can edit the transaction if the transaction is still processing. But he cannot hold or manage that transaction.

Authenticator

Authenticator is derived from stakeholders when the creating the transaction for sign and Authenticate the legitimacy of particular transaction for successful transaction.

De-Authenticator

De-Authenticator is also derived from stakeholders by himself when the stakeholder De-Authenticate the transaction and also he become an authenticator for farther proof legitimacy of authentication transparency.

Other User

Any of the user access the system he can only view status and can be audited of the process.

Ethereum Blockchain

The Ethereum Blockchain is the underground technologies uses to automatize the processes, store data and securing the ledger of the Decentralized Secure Transparent prototype system.

3.3 System Architecture

The architecture of the Decentralized Secure Transparent (prototype) System is design with frontend and backend process which shown in figure : ??, the main components of the architecture are in brief;

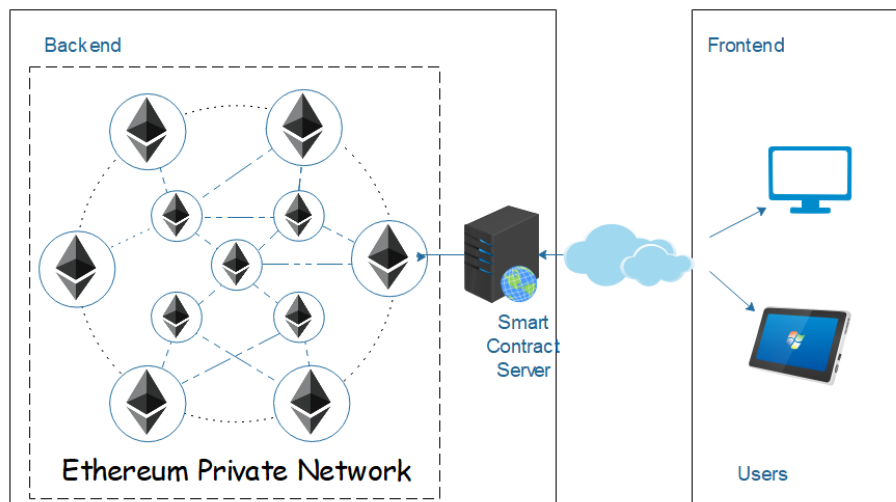


Figure 3.3: The architecture of the Decentralized Secure Transparent (prototype) System

Frontend

The DApp is the frontEnd Application is what users willing to access the system via internet.

- Desktop / Tablet device

End Devices that are used to end-users for creating, managing, Authenticating and viewing the status of_ transactions which exchanging among stakeholders.

Backend

The backend is a Blockchain based system which directly communicate with Ethereum network for the supplied serviced to frontend client consumes.

- Smart Contract Server

The server hosts an Ethereum node and Smart contract Application that is part of the Ethereum blockchain network.

- Ethereum network

Ethereum network consists of many nodes that are used to record transactions. Therefore it can be seen as a public digital-ledger. After validation a transaction is entered in to a block and it can never be altered thereafter. Consequently participants can verify and audit transactions by themselves at relatively cost-effective prices.

3.4 Tools Used

- **E.V.M**

EVM stands for Ethereum Virtual Machine. It provides a run time environment based on ethereum for smart contract execution and mining blocks of the blockchain.

- **Solidity**

Smart contracts are usually written in solidity. Solidy is a very popular programming language in the ethereum block chain. It has almost beome the official language for ethereum development. It was designed specifically for ethereum virtual machine.

- **Remix ethereum - IDE**

Remix IDE is a compiler and runtime environment. It is run on a browser and is also open-source. This is used to write solidity or Vyper smart contracts. Written in Java script, this IDE, performs debugging, testing and deploying smart contacts.

- **MetaMask**

Metamask is a browser based ethereum extension tool for Firefox, chrome, and opera. It provides a simpler platform for users to interact with the ethereum blockchain. This is a secure platform that users utilize for managing their identities and signing their blockchain transactions.

- **Web3**

Web 3.0 is a more user centric blockchain stack that has a decentralized network. This is a more transparent and secure internet that is attempting to create a more user friendly apps.

Chapter 4

IMPLEMENTATION

The specific details and design decisions regarding the disentailed secure transparent system are described in this chapter. According to the system architecture discussed in Section 3.2 as shown in the Figure: 4.1 in page no 31, the implementation is mainly divided into two parts as backend and frontend. The backend is a Blockchain based application which directly communicate with Ethereum network to provide the necessary services for frontend clients. The frontend requests information from the backend process and displays it to the user. The front end embed the public and private keys for the user.

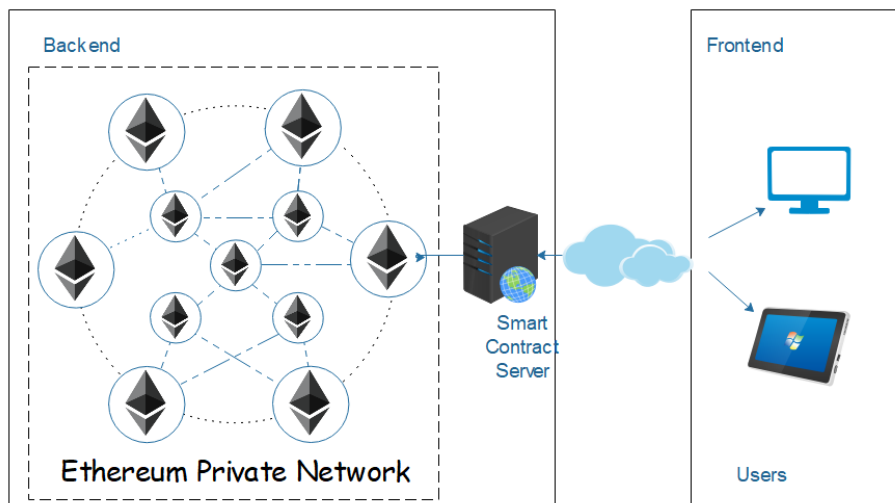


Figure 4.1: The architecture of the Decentralized Secure Transparent (prototype) System reference from page no 31

4.1 Frontend

The DApp is the frontend Application is what users willing to access the system via internet. DApp is browser based device independent application which is directly interact with Metamask or

any decentralized client app. Which hold the Ethereum wallet of clients (privet and public keys). The researcher used metamask to communicate with the private ethereum block chain and create a wallet that was used to access the application. This was due to the fact that Meta mask is a browser extension for chrome, Firefox and opera. It is easy to install and interact with the system.

4.2 Backend

The backend is a based on the blockchain and related technologies, which directly communicate with Ethereum privet network for the supplied serviced to frontend client consumes which is mainly consist with of three major subsystems.

1. Ethereum Private Network.
2. Smart Contract Application
3. Smart contract server

4.2.1 Ethereum Private Network

According to the amount of computation the network provides to execute a smart contract the user need to pay a price. This price which is paid in ether is known as gas. The cost of gas may be too high for running on complex smart contract. The main Ethereum network gas is obtainable, it involves purchasing and depositing actual ETH into the account. So for testing and development purposes researcher used a type of simulation technique for blockchain which is ideal platform for testing blockchain based apps with lesser cost and resources. And also Ethereum Private Network is a recommended platform for a deployment of the real system due high gas price. This price usually increases with the increase of the system complexity.

When nodes are not linked to the main ethereum network it is called a private network. Therefore this is reserved and isolated but does not necessarily mean it's protected or secured. The security, immutability, and reliability of such private ethereum networks can be improved by increasing the number of nodes and servers.

There's a several ways to deploy an ethereum private network. This test network consist with one physical server and one ethereum node but if more nodes and servers are needed we can increases the no of servers. Which is more reliable and much more secure. The simplest way to development and test Decentralized Secure Transparent System the researcher uses Ganache GUI as a personal ethereum blockchain network for Ethereum Development which is installed on Ubuntu Linux server. If much more security is needed for more reliable mechanism the researcher recommends Geth framework.

4.2.1.1 Installing Ganache GUI

Ganache GUI shown is figure :4.2. Is used for creating the personal ethereum blockchain. This is what the researcher used for ethereum development and deploy smart contracts, develop applications and test smart contracts. To install ganache GUI it can be directly download by visiting the home page of Ganache. There are specific formats for Windows, Linux and Mac as well.

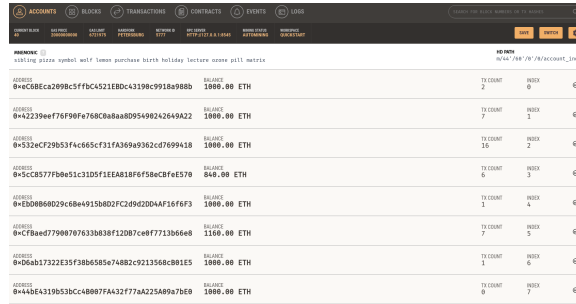


Figure 4.2: Ganache Personal Blockchain for Ethereum Development

4.2.1.2 Install Metamask and Integrate with Ganache GUI

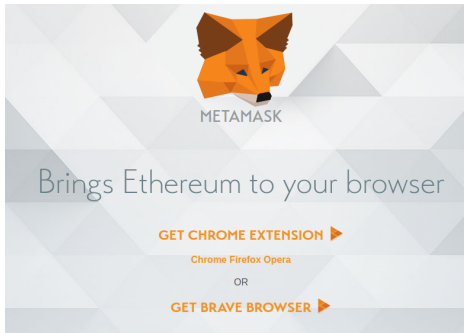
To access the blockchain, the user needs to install a mobile distributed application (DApp) browser. So the most popular DApp browsers is Metamask which is a google chrome, firefox, opera extension. Therefore it is a prerequisite to install one of the above mentioned web browser as well. A brief description of how to install metamask for chrome is given bellow.

Step 1 : Go to the Metamask website using google chrome browser.

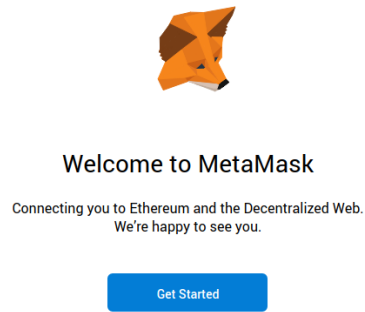
Step 2 : Click on “Get Chrome Extension” to install Metamask, as seen in 4.3a.

Step 3 : Click “Add to Chrome” in the upper right corner.

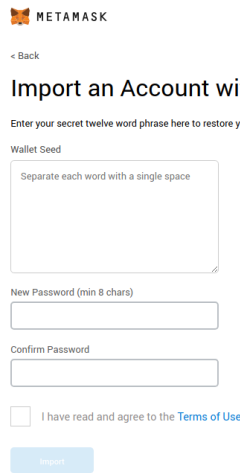
Step 4 : Click “Add Extension” to complete the installation.



(a) Installing Metamask extension



(b) Configuring Meta mask



(c) Import the wallets

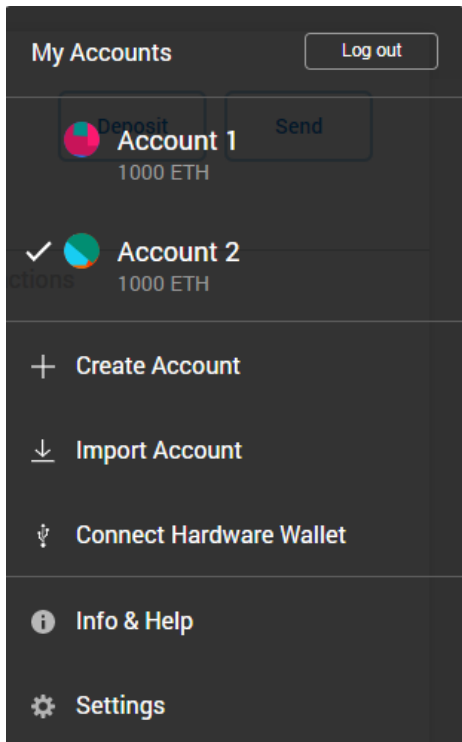
Figure 4.3: Install Metamask and Integrate with Ganache GUI

Step 5 : After Metamask is successfully installed click “Get Started” as shown in 4.3b.

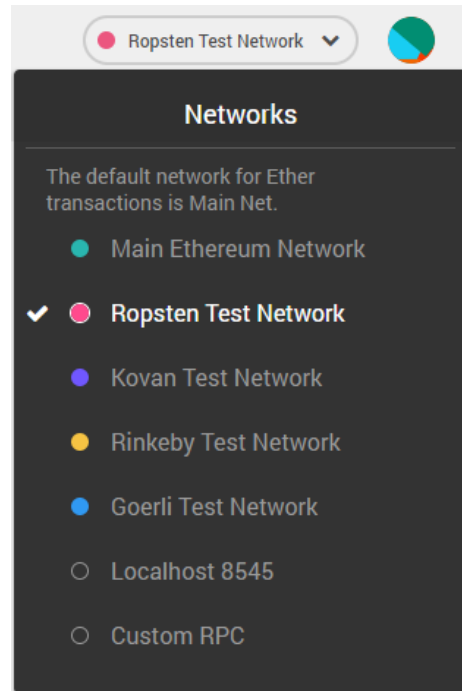


Figure 4.4: Mnemonic Seed Words on ganache GUI

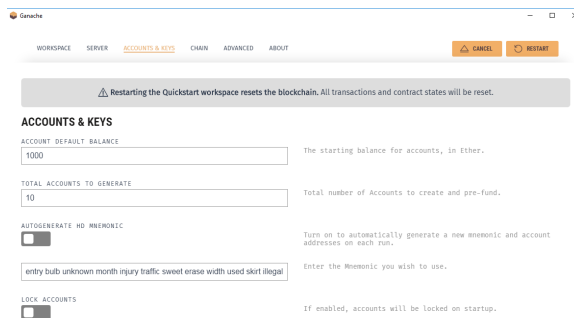
Step 6 : Then click ”Import Wallet” followed by “I agree”. Copy and paste Mnemonic Seed Words on ganache as shown in GUI 4.3c into wallet seed and set password of the account. Then click “Import” and “All Done”



(a) Add Accounts



(b) Set Network



(c) Gnache GUI Sttings

Figure 4.5: General Settings on Ganache and Metamask!

Now set the network (Refer Figure : 4.5b) to 'Localhost 8545' or settings what you provided earlier to ganache. Now you have connected to metamask and ganache GUI with your Account. If you need to increase the no of accounts you can add (Refer Figure 4.5a). So the ganache prop-up more account to you according to the relevant settings (Refer Figure 4.5c that you set in the ganache GUI).

4.2.2 Smart Contract Application

The Decentralized Secure Transparent System is purely blockchain-based application. All of the data flows regarding the request from this System (smart contract), is stored in Ethereum Blockchain. The Smart contract is developed in solidity (Refer 2.9.7) language. The development and testing with Ethereum Remix environment (Refer 2.9.6) and Ethereum development framework Truffle. Truffle framework will help to the creating the essential files when creating the Dapp.

The full source code of the Transparent-System smart contract will be post as an Appendix-A and some of the data stores, major functionalities and function modifiers on the contract described in below.

4.2.2.1 Function Modifiers

Function modifiers can change the behavior of a function also validate the inputs that means modifiers are able to check a given condition prior to running the function. In this smart contact the researcher use modifiers to validate the authority and to access particular functions.

Only_Admin Modifier

“Only_Admin” modifier is used to check if the person sending the function request is an admin or not. This is done by checking if the public key of the sender is equal to the public key of the admin.

```
modifier Only_Admin {
    require(msg.sender == Admin);
    -;
}
```

Only_Creater Modifier

“Only_Creater” modifier is used to check if the person sending the function request to change a transaction is the creator of the same transaction or not. This is done by checking if the public key of the sender is equal to the public key of the creator of the transaction.

```
modifier Only_Creater(uint TId) {
    require(msg.sender == Transactions[TId].Creator);
    -;
}
```

Only_Stakeholders Modifier

“Only_Stakeholders” modifier is used to check if the person sending the function request is a stakeholder or not. This is done by checking through the list of stakeholders if the public key of the sender is equal to the public key of one of the stakeholders.

```
modifier Only_Stakeholders{
    for(uint J = 0;J < Stakeholders.length; J++){
        if(Stakeholders[J].Stakeholder==msg.sender){
            -;
            return;
        }
    }
}
```

```

    }
    }
    revert();
    -;
}

```

Only_Authenticator Modifier

“Only_Authenticator” is used to check if the person sending the function request is a authenticator for that transaction or not. This is done by checking through the list of authenticators if the public key of the sender is equal to the public key of one of the authenticators

```

modifier Only_Authenticator(uint TId){
    uint x= Transactions[TId].Authenticators.length;
    for(uint i = 0;i < x ; i++){
        if(Transactions[TId].Authenticators[i].Authenticator==msg.sender){
            -;
            return;
        }
    }
    revert();
    -;
}

```

4.2.2.2 Data stores

According to the controls required to fulfil the requirements of legal framework in government sector and auditing procedures. Researcher identified some of basic criteria to be considered when designing the prototype system. The system must keep recodes of every transactional data, list of Authenticators and De-authenticators. Each and every structures keeps necessary details which needs to be audited further. It is defined in the code bellow:

Stakeholder Data Store

The stakeholder structures keeps multiple details to identify the stakeholders.

```

struct Stakeholder{
    address Stakeholder;
    string Name;
    string Designation;
    uint Stakeholder_Since;
}

```

Stakeholder array helps to add multiple stakeholders to the system.

```
Stakeholder[] public Stakeholders;
```

Authenticator Data Store

The Authenticate structures keeps multiple details to identify the Authenticator and his comments. Authentication_1st Authentication_2nd and Authentication_3rd is holding the acceptance details put in by the Authenticator because any transaction can only be authenticated 3 times. If the transaction is de-authenticated after the third time the transaction will be halted. These 3 variables are used to hold the comments of approval by the authenticator.

```

struct Authenticate{
    address Authenticater;
    address Added_By;
    bool isApproved;
    string reson_To_Added;
    string Authentication_1st;
    string Authentication_2nd;
    string Authentication_3rd;
}

```

De_Authenticator Data Store The De_Authenticate structures keeps multiple details to identify the De_Authenticater and his comments.

```

struct De_Authenticate{
    address De_Authenticater;
    string De_AuthReason;
}

```

Transaction Data Store

The Transaction structures keeps multiple details to identify the Transaction. The details of the transaction should be entered correctly for further auditing purpose by the transaction creator. That is a organization policy that the users should maintain. This structure is keeps Authenticate and De_Authenticate list for each transaction for farther auditing and the TransactionHash provides the integrity for the transaction.

```

struct Transaction {
    string Name;
    string Description;
    string Condition;
    address Creator;
    address Recipient;
    uint amount;
    bool Stop_Transaction ;
    bytes32 TransactionHash;
    Authenticate [] Authenticaters;
    De_Authenticate [] De_Authenticaters;

}

```

Transaction array help to add multiple Transactions to the system

```
Transaction[] public Transactions;
```

4.2.2.3 Functionalities

Change System Administrator

A request to change the administrator can only be done by the admin. This function checks if the sender of the request is the admin using the Only_admin modifier before assigning a new admin. To view the results of this process please refer section 5.0 .

```
function Change_Admin(address New_Owner) Only_Admin public {
    Admin = New_Owner;
}
```

Add Project Stakeholders

Project stakeholders are those with any interest in the project's outcome, such as project managers, customers, contractors, executives, project sponsors as well as Auditors and also any other key individuals. Project Administrator is the only party who can add the stakeholders and remove them from the system. Therefore this function checks if the sender of the request is the admin using the Only_admin modifier. However before the request can be sent the admin needs to enter the public key of the stake holder. So every stakeholder has to generate a wallet and send the public key to the admin before the admin could request this function.

```
Function Add_Stakeholder (address Target_Stakeholder, string Stakeholder_Name,
string Stakeholder_Designation) Only_Admin public {

    uint id = Stakeholder_Id[Target_Stakeholder];
    if (id == 0) {
        Stakeholder_Id[Target_Stakeholder] = Stakeholders.length;
        id = Stakeholders.length++;
    }

    Stakeholders[id] = Stakeholder({
    Stakeholder: Target_Stakeholder,
    Stakeholder_Since: now,
    Name: Stakeholder_Name,
    Designation:Stakeholder_Designation  });

    emit Stakeholder_Changed(Target_Stakeholder, true);
}
```

Remove Project Stakeholders

The admin can remove stakeholders from the system by using the stakeholder's public key. If someone is not a stakeholder of the project any more this is necessary for security and confidentiality concerns.

```
function Remove_Stakeholder(address Target_Stakeholder) Only_Admin public {
    require(Stakeholder_Id[Target_Stakeholder] != 0);

    for (uint i = Stakeholder_Id[Target_Stakeholder]; i<Stakeholders.length-1;
        i++){
        Stakeholders[i] = Stakeholders[i+1];
    }
    delete Stakeholders[Stakeholders.length-1];
    Stakeholders.length--;
}
```

Create Transactions

Project stakeholders are the only authorised party to create transaction. when creating the transaction creator must add authenticators (list of other stakeholders) public address for verification of the transaction. if one member among them will de-authenticate the transaction, others must authenticate another time such that this process can iterate only three times then the transaction will be blocked and it will no further proceed. This is what helps to prevent fraud. auditors will be alerted of such transactions.

```
function newTransaction( string Tra_Name, string Tra_Condition, string
    Tra_Description, address Next_Recipient, address[] Authenticators_List )
    Only_Stakeholders public payable returns (uint TransactionID)
    {
        TransactionID = Transactions.length++;
        Transaction storage p = Transactions[TransactionID];
        p.Name = Tra_Name;
        p.Condition=Tra_Condition;
        p.Description = Tra_Description;
        p.Creator = msg.sender;
        p.Recipient = Next_Recipient;
        p.amount= msg.value;
        p.TransactionHash = keccak256( Tra_Name, Tra_Condition, msg.sender,
            Next_Recipient, msg.value, now);
        p.Stop_Transaction = false;

        for(uint256 i=0; i<Authenticators_List.length; i++)
        {
            New_Authenticator( TransactionID, Authenticators_List[i], msg.sender, "
                Initial Authenticators");
        }

        New_Authenticator( TransactionID, msg.sender, msg.sender, "Initial
            Authenticators");
        New_Authenticator( TransactionID, Next_Recipient, msg.sender, "Initial
            Authenticators");
        numTransactions = TransactionID+1;
        return TransactionID;
    }
}
```

Edit Created Transaction

Only the creator can edit a transaction. This is ensured by only_creator modifier. This useful if the creator made a mistake during his request. If the amount was entered incorrectly the difference will be deposited back to the creator's wallet.

```
function Edit_Transaction( uint Transaction_ID, string Tra_Name, string
    Tra_Condition, string Tra_Description, address Next_Recipient, address[]
    Authenticators_List ) Only_Creator(Transaction_ID) public payable
    {
        Transaction storage x = Transactions[Transaction_ID];
        x.Creator.transfer(Transactions[Transaction_ID].amount);
        x.Name = Tra_Name;
        x.Condition=Tra_Condition;
        x.Description = Tra_Description;
        x.Creator = msg.sender;
        x.Recipient = Next_Recipient;
        x.amount= msg.value;
    }
}
```

```

x.TransactionHash = keccak256( Tra_Name,Tra_Condition , msg.sender,Next_Recipient
,msg.value,now);
x.Stop_Transaction = false;

for(uint256 i=0;i<Authenticaters_List.length;i++)
{
    New_Authenticator( Transaction_ID, Authenticaters_List[i] ,msg.sender,"
    Initial Authenticaters");
}
    New_Authenticator( Transaction_ID, msg.sender,msg.sender,"Initial
    Authenticaters");
    New_Authenticator( Transaction_ID, Next_Recipient,msg.sender,"Initial
    Authenticaters");
}

```

Authenticate the Transactions

Transactions are approved only by the authenticators. This is ensured by the only_authenticator modifier. The authenticator needs to input the transaction ID to the system before authenticating and if any stakeholder de-authenticated the transaction then the authenticator has to authenticate again. In such cases the authenticator may have to authenticate up to 3 times. If the transaction is de-authenticated 3 times the transaction will freeze and the creator will receive his cash back. If all the authenticators have approved the transaction then the transaction will be completed and receiver will receive the cash.

```

function approve(uint TracID,string _Comments) Only_Authenticator(TracID) public {
    bool isAllApproved=true;
    for(uint256 i=0;i<Transactions[TracID].Authenticaters.length;i++)
    {
        if(Transactions[TracID].Authenticaters[i].Authenticator == msg.sender)
        {
            Transactions[TracID].Authenticaters[i].isApproved = true;
            if (Transactions[TracID].De_Authenticaters.length==0){
                Transactions[TracID].Authenticaters[i].Authentication_1st =
                _Comments;
            }
            if (Transactions[TracID].De_Authenticaters.length==1){
                Transactions[TracID].Authenticaters[i].Authentication_2nd =
                _Comments;
            }
            if (Transactions[TracID].De_Authenticaters.length==2){
                Transactions[TracID].Authenticaters[i].Authentication_3rd =
                _Comments;
            }
        }
        if(!Transactions[TracID].Authenticaters[i].isApproved)
            isAllApproved = false;
    }
    if(isAllApproved && !Transactions[TracID].Stop_Transaction)
    {
        Transactions[TracID].Recipient.transfer(Transactions[TracID].amount)
        ;
    }
}

```

Add Authenticator to the Transactions

If an authenticator needs to add another authenticator they can do so using this function. The only_authenticator modifier guarantees that the person adding another authenticator is already an authenticator of the transaction. The person being added as an authenticator needs to be a stakeholder.

```
function Add_Authenticator(uint TransactionID, address New_Authenticator_Address,
string Purpose) Only_Authenticator(TransactionID) public {
    New_Authenticator( TransactionID, New_Authenticator_Address ,msg.sender,
    Purpose);
}

function New_Authenticator(uint TransactionID, address
New_Authenticator_Address, address Add_By, string Reson_To_Added) private{

    uint Authenticater_id = Transactions[TransactionID].
    Authenticaters.length++;
    Authenticate storage a =Transactions[TransactionID].Authenticaters[
    Authenticater_id];
    a.Authenticater =New_Authenticator_Address;
    a.isApproved = false;
    a.Added_By=Add_By;
    a.reson_To_Added= Reson_To_Added;

}
}
```

De-Authenticate the Transactions

A transaction can be de-authenticated by any stakeholder. The only_stakeholder modifier is used to guarantee that the person de-authenticating the transaction is a stakeholder. If the transaction is de-authenticated 3 times the transaction will freeze and the creator will receive his cash back. This happens due to the change in stop_transaction variable from false to true.

```
function De_Authenticates(uint8 TracID, string De_Auth_Reason, string Comments)
Only_Stakeholders public returns(uint) {

    for(uint256 i=0; i<Transactions[TracID].Authenticaters.length; i++)
    {
        Transactions[TracID].Authenticaters[i].isApproved = false;
    }

    New_Authenticator( TracID, msg.sender, msg.sender, Comments);
    uint De_Authenticator_id = Transactions[TracID].De_Authenticaters.
    length++; De_Authenticator to de-authanticates list for ferther
    knolage
    De_Authenticate storage d =Transactions[TracID].De_Authenticaters[
    De_Authenticator_id];
    d.De_Authenticator = msg.sender;
    d.De_AuthReason = De_Auth_Reason;

    if (De_Authenticator_id >=2){
        Transactions[TracID].Stop_Transaction=true;
        Transactions[TracID].Creator.transfer(Transactions[TracID].amount);
        Transactions[TracID].amount =0;
    }

}
```

4.2.3 Smart Contract Server

The server hosts an Ethereum node and Smart contract, Decentralized Application that is part of the Ethereum blockchain network. The DApp is the frontend Application. It is what users will access. Instead of being centered on a central server, distributed applications spread throughout multiple servers. This is useful at times of huge data and traffic demand because it prevents a system breakdown. This is especially useful for critical projects where downtime is not an option.

But for testing purposes the researcher is using only one physical server to host the prototype Dapp. To develop the Decentralized Secure Transparent System researcher used a single Ubuntu Linux server as a personal blockchain network for Ethereum Development.

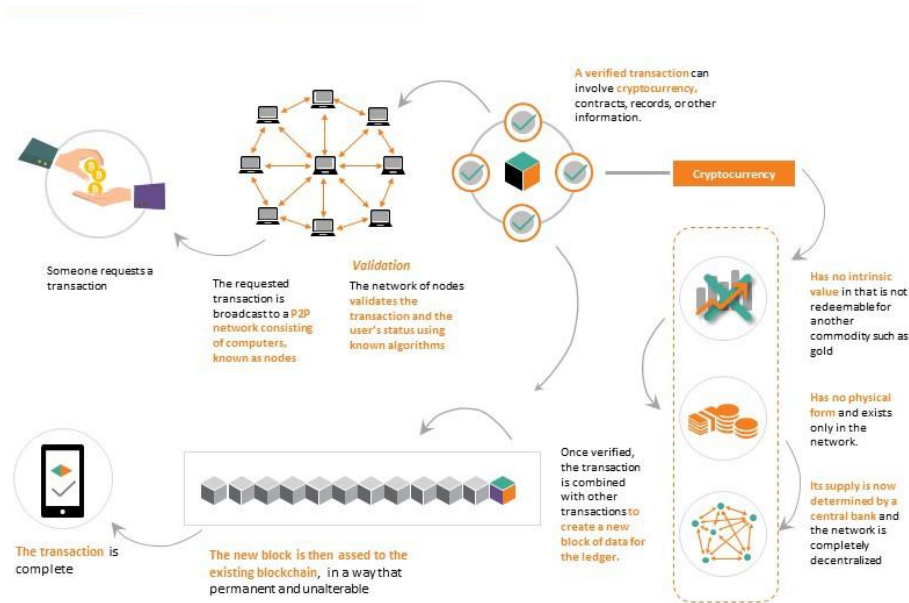


Figure 4.6: Dapp portfolio

4.2.3.1 Deployment of the Decentralized Secure Transparent (Prototype) System

Step 01 :

To deploy the prototype system researcher use "Remix - Solidity IDE" (Refer Figure :4.7.) which is the most common and powerful tool for testing and developing smart contracts. And make sure to run ganache and metamask in backend (Refer Subsection 4.2.1.2).

Step 02 :

copy and paste solidity code into the development background and go to the compile tab and set the compiler version as "0.4.22+commit.4cb486ee" so that when it successful it can be seen with no errors as in Figure 4.6.

Step 03 :

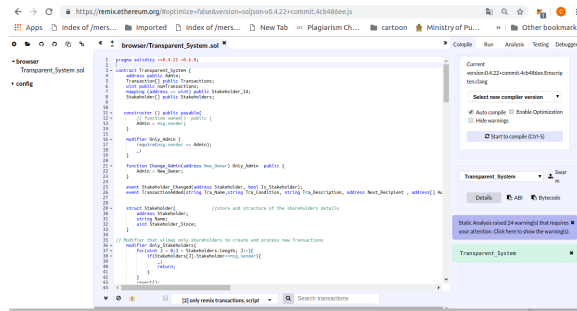


Figure 4.7: Remix - Solidity IDE

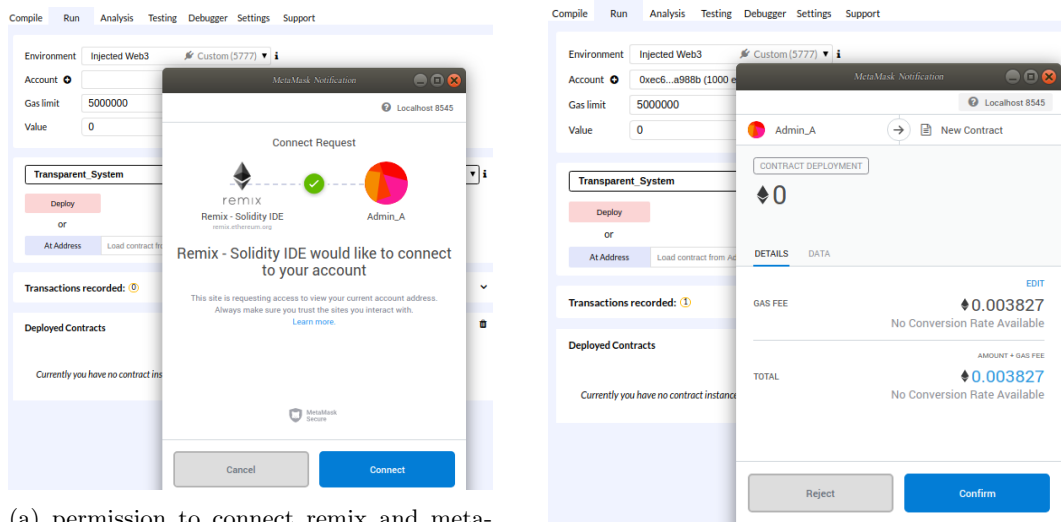
To deploy the system user needs to set the Environment. go to the Run tab and set the Environment to Injected web3 then it will ask permission to connect remix and metamask (Refer the Figure 4.7a).

Step 04 :

Set the Gas limit as 5000000 and click deploy. the system it will pop-up window again ask for confirmation for the transaction deployment approval from the particular account holder (Refer the Figure : 4.7b).

Step 05 :

So after the successful Deployment the system shown as Figure : 4.8



(a) permission to connect remix and meta-mask

(b) confirmation for the transaction deployment

Figure 4.8: System Deployment

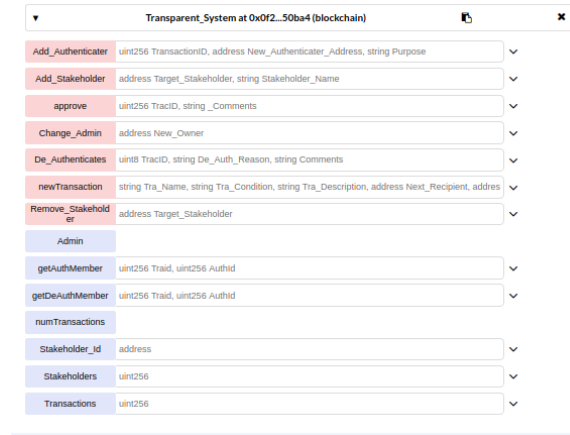


Figure 4.9: Decentralized Secure Transparent (Prototype) System

Metamask information about the deployment of the Decentralized Secure Transparent (Prototype) System is shown. It includes account holders transaction metadata and the account balance.

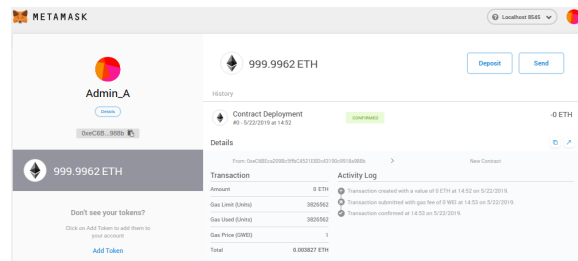


Figure 4.10: Metamask informations for deployment

Ganache GUI also shows some of the information regarding the transaction which transaction data is encrypted.

The screenshot shows the Ganache interface with a transaction details view. The transaction hash is `TX 0x1e355714b13642072a6f19d737ea16128665fd38b724d5cf4760025fb5a961ca`. Below the hash, the sender address is `0xeC6BEca2098c5ffbC4521E8DC43198c9918a988d` and the created contract address is `0x0f29f37f553fc0dA9065f833bA342390c7F50ba4`. A table below shows the transaction details:

VALUE	GAS USED	GAS PRICE	GAS LIMIT	MINED IN BLOCK
0.00 ETH	3826562	100000000	3826562	1

At the bottom, there is a 'TX DATA' section with a long hexadecimal string representing the transaction data.

Figure 4.11: Ganache informations for deployment

Chapter 5

RESULTS AND CONCLUSION

Initially, ten Ethereum accounts were created with 1000 ethers on the Ethereum private blockchain. Two accounts were named as Admin-A and Admin-B and the other accounts are used to project stakeholder address.

Change System Administrator

Get the proposed Administrators public address and then transfer the system Administration to new administrator. Function description 4.2.3 section..

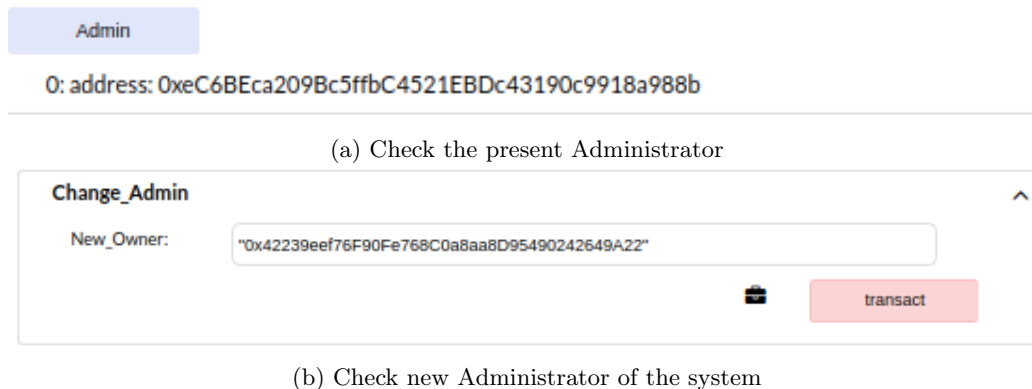


Figure 5.1: Sign the request and check new Administrator

Figure: 5.1a shows that check present administrator and also Figure: 5.1b shows that the Transferring to Administration to new Administrator by his public key.

Figure : 5.3 shows Approving and signing the transaction and Figure :5.1b shows that new Administrator in the system

Admin

0: address: 0x42239eef76F90Fe768C0a8aa8D95490242649A22

Figure 5.2: Check new Administrator of the system

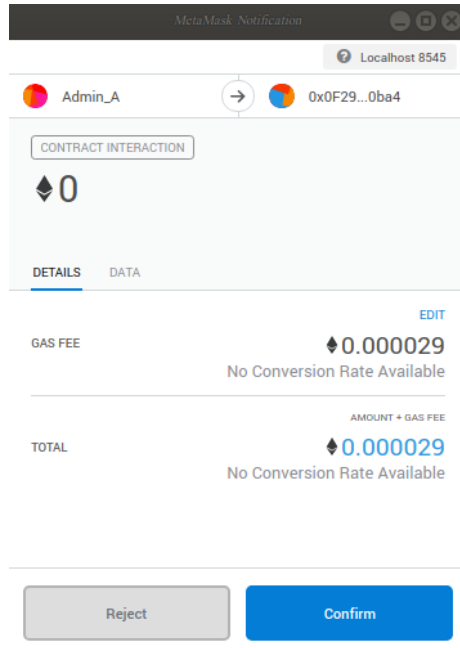


Figure 5.3: Sign and accept the Request by present Admin

Add Project Stakeholders

Administrator is the only Authorised party to add stakeholder(Refer Figure :??. for the function description please refer 4.2.3 section.

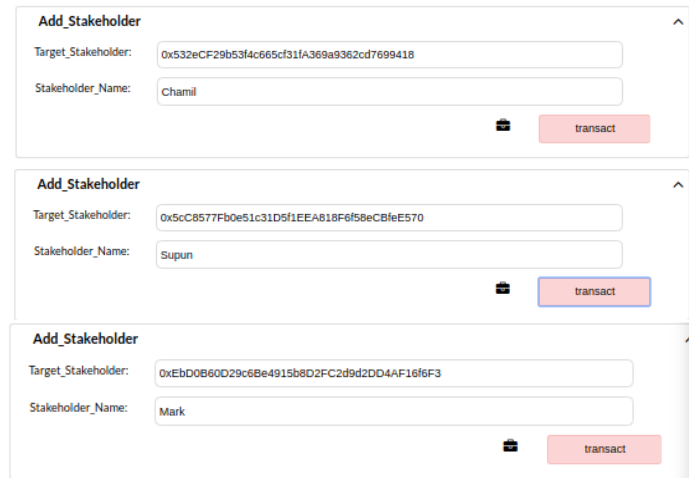
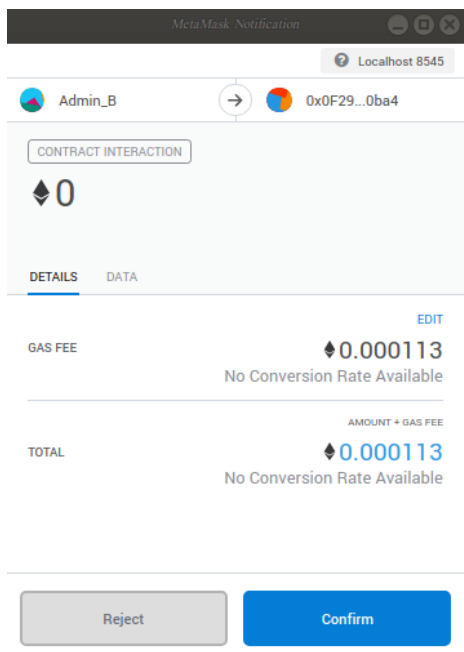
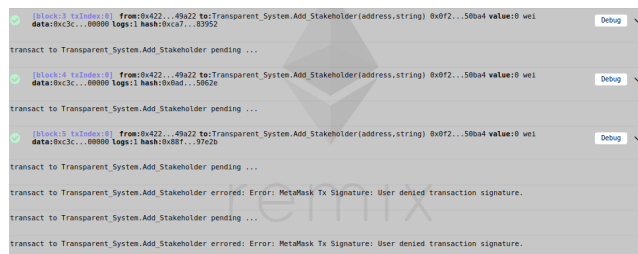


Figure 5.4: Add Project Stakeholders by Administrator

Each time Administrator has to sign using his private key (Refer Figure 5.5a) otherwise stakeholder will not be added to the system. And the Figure 5.5b shows 3 stakeholders have been added to the system and 2 have been rejected by administrator(Remix-output)



(a) Administrator sign by his private key



(b) Add Project Stakeholders Remix-informations

Figure 5.5: Add Project Stakeholders by Administrator

Create Transactions

Transactions can only be created by stakeholders. For the function description please refer 4.2.3 section. (Refer in Figure 5.6).

The figure displays two examples of the 'newTransaction' form. Each form contains the following fields:

- Tra_Name:** Cisco switches (top) / "IBM Servers" (bottom)
- Tra_Condition:** delivered before 30 days -5 units| 26x1000+2xSFP -L3 - Flash : 16 MB Switching : 56 Gbps (top) / "delivered before 30 days -4 units| must be match specs sheet : https://www-07.ibm.com/servers/eserver/includes/content/pseries/downloads/facts" (bottom)
- Tra_Description:** network wiring equipments to Galle brunch (top) / "network Server to head office" (bottom)
- Next_Recipient:** 0xCfBaed77900707633b838f12DB7ce0f7713b66e8 (top) / "0xCfBaed77900707633b838f12DB7ce0f7713b66e8" (bottom)
- Authenticaters_List:** ["0xD230CF84c5d4edfd02aCB99f4Adb1A3e11ac03B5", "0x38478580f3e5FDe6479cd11D004835d86E"] (top) / ["0x5c8577Fb0e51c31D5f1EEA818F6f59eCBfeE570", "0xD6ab17322E35f38b6585e74882c9213568cB01E5"] (bottom)

Each form includes a 'transact' button at the bottom right.

Figure 5.6: Create Transactions by Any Stakeholder

Then each time transaction creator must sign with his private key using Meta mask. Please refer the Figure : 5.7. .

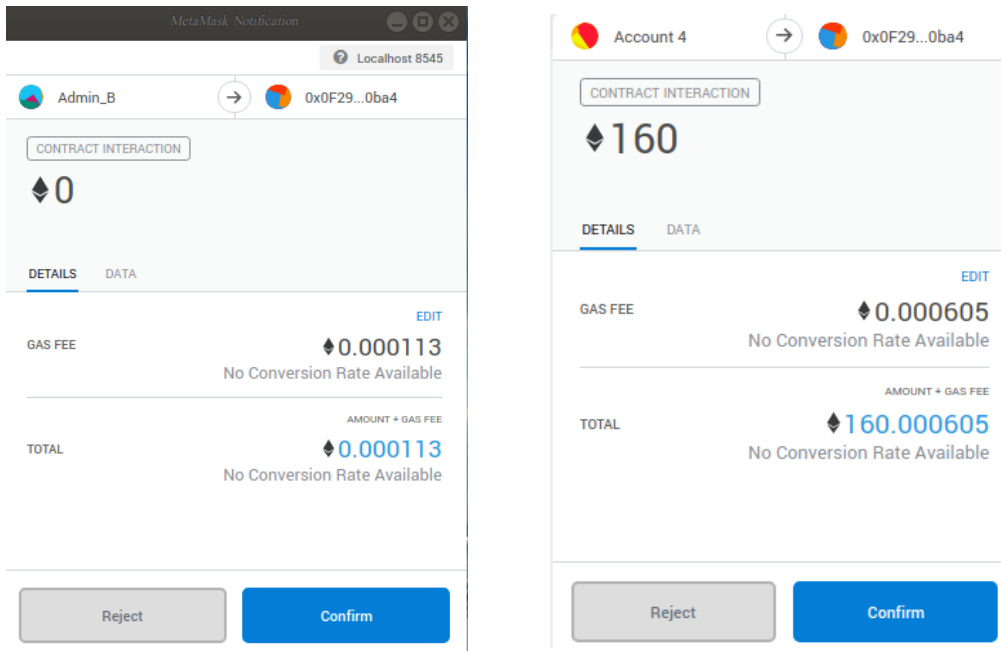


Figure 5.7: Transaction Creator sign transaction by his private key

Transactions de-authentication by stake holder

Transaction created by 0xCfBaed77900707633b838f12DB7ce0f7713b66e8 (Account 3) for transfer 140 ethers. please refer figure : 5.8 and you can see the ether deduction on the transaction creator's wallet figure 5.9.

newTransaction

Tra_Name: "IBM Servers"

Tra_Condition: "delivered before 30 days -4 units] must be match specs sheet : https://www-07.ibm.com/servers/eserver/includes/content/series/downloads/facts"

Tra_Description: "network Server to head office"

Next_Recipient: "0xCfBaeD77900707633b83812D87ce07713b66e8"

Authenticators_List: ["0x5c8577F0e51c31D5fEEA818f8eC8fE570", "0xD6ab17322E35f38b6585e748B2c9213568cB01E5"]

transact

Environment: Injected Web3 Custom (5777)

Account: 0x532...99418 (999.999380141 ether)

Gas limit: 5000000

Value: 140 ether

Figure 5.8: Transaction created for transfer 140 ethers.

Account 3 → 0x0F29...0ba4

CONTRACT INTERACTION

140

DETAILS DATA

GAS FEE 0.000608 No Conversion Rate Available

TOTAL 140.000608 No Conversion Rate Available

Reject Confirm

(a) sign transaction for further process

Account 3

0x532e...9418

859.9963 ETH

(b) deduct the ethers from Creators wallet

Figure 5.9: Sign the Transaction for transfer 140 ethers.

As shown in figure :5.10 the 7th point "Stop-Transaction" is "false". That means transaction can be further processed. If stakeholders de-authenticate the Transaction by three times (Refer Figure

```

0: string: Name IBM Servers
1: uint256: Time_Stamp 1558524445
2: string: Description network Server to head office
3: string: Condition delivered before 30 days -4 units| must be match specs sheet : https://www-07.ibm.com/servers/eserver/includes/content/pseries/downloads/factsfeatures.pdf
4: address: Creator 0x532eCF29b53f4c665cf31fA369a9362cd7699418
5: address: Recipient 0xCfBaed77900707633b838f12DB7ce0f7713b66e8
6: uint256: amount 14000000000000000000
7: bool: Stop_Transaction false

```

Figure 5.10: Created Transaction details

: 5.11) then the "Stop-Transaction" parameter has changed to "true". So transaction cannot be process any further. Amount of ether 140 has been send back to the Transaction creator (account 3) refer Figure: 5.13a and also shows "Account 3" stake-holder's logs where can one failed transaction can be seen. Please refer to figure :5.13b.

getDeAuthMember ^

Traild:

AuthId:

call

0: address: Authenticators 0x532eCF29b53f4c665cf31fA369a9362cd7699418
1: string: De_Auth_Reason no deal power
2: uint256: dAuthnums 2

Figure 5.11: De-Authenticate the Transaction

Transactions v

0: string: Name IBM Servers
1: uint256: Time_Stamp 1558524445
2: string: Description network Server to head office
3: string: Condition delivered before 30 days -4 units| must be match specs sheet : https://www-07.ibm.com/servers/eserver/includes/content/pseries/downloads/factsfeatures.pdf
4: address: Creator 0x532eCF29b53f4c665cf31fA369a9362cd7699418
5: address: Recipient 0xCfBaed77900707633b838f12DB7ce0f7713b66e8
6: uint256: amount 0
7: bool: Stop_Transaction true

Figure 5.12: De-Authenticated transaction

Successful transaction

Transaction created by Account 4 for transferring 160 ethers. please refer figures num : 5.14 and 5.15.

Environment: Injected Web3 Custom (5777) *i*

Account: 0x5cc...ee570 (999.999746055 ether) *📄* *✎*

Gas limit: 5000000

Value: 160 ether

newTransaction

Tra_Name: cisco switches

Tra_Condition: delivered before 30 days -5 units| 26x1000+2xSFP -L3 - Flash : 16 MB Switching : 56 Gbps

Tra_Description: network equipment to Gall brunch

Next_Recipient: 0xCf8aed77900707633b38f12DB7ce0f7713b66e8*

Authenticators_List: [0xEbd0860D29c6Be4915b8D2FC2d9d2DD4AF16f6F3*, 0xD6ab17322E35f38b6585e748B2e9213568cB01E5*]

📄 **transact**

Figure 5.14: Transaction created for transfer 160 ethers...

Account 4 → 0x0F29...0ba4

CONTRACT INTERACTION

160

DETAILS DATA

GAS FEE: 0.000605 *EDIT*
No Conversion Rate Available

TOTAL: 160.000605
No Conversion Rate Available

Reject Confirm

Account 4
Details
0x5cCB...E570 *📄*

839.9991 ETH

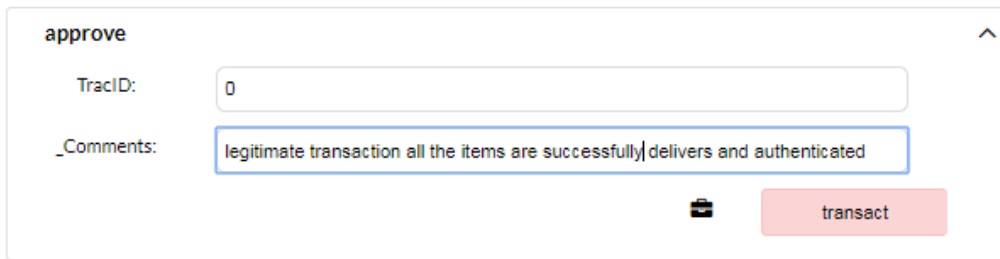
Figure 5.15: Transaction creator approved for transfer 160 ethers...

Please refer to the Figure : 5.16a where in the 7th point "Stop-Transaction" is "false". so that means transaction can be further proceed. if all authenticators has approved the transaction (please

refer figure : 5.16b) then it must be a legitimate transaction. so the requested amount of ether 160 has been send to Recipient's (refer Figure: 5.16) wallet.

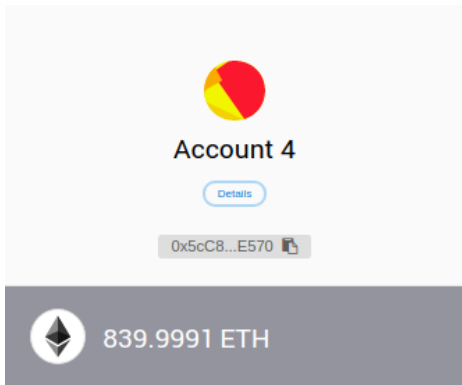
```
0: string: Name cisco switches
1: uint256: Time_Stamp 1558526670
2: string: Description network equipment to Gall brunch
3: string: Condition delivered before 30 days -5 units| 26x1000+2xSFP -L3 - Flash : 16 MB Switching : 56 Gbps
4: address: Creator 0x5cC8577Fb0e51c31D5f1EEA818F6f58eCBfeE570
5: address: Recipient 0xCfBaed77900707633b838f12DB7ce0f7713b66e8
6: uint256: amount 1600000000000000000000
7: bool: Stop_Transaction false
```

(a) Transaction Details.

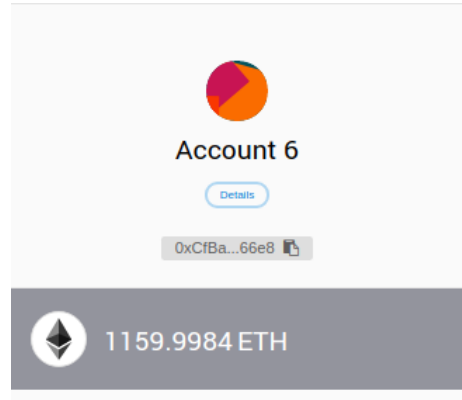


(b) Transaction appovements

Figure 5.16: Transaction approved for transfer 160 ethers...



(a) Transaction Senders wallet



(b) Transaction Recipient wallet

Figure 5.17: Transaction approved for transfer 160 ethers...

All users can seen the status of the transaction such as Pease refer figures on 5.18

getAuthMember

Traid:

AuthId:



0: address: Authenticator 0xD6ab17322E35f38b6585e748B2c9213568cB01E5
 1: bool: isApproved true
 2: string: Authentication_1st Accepting to transfer money this seems to be a legitimate transaction
 3: string: Authentication_2nd
 4: string: Authentication_3rd
 5: address: Added_By 0x5cC8577Fb0e51c31D5f1EEA818F6f58eCBfeE570

(a) Check Authenticators status

getDeAuthMember


Traid:

AuthId:



0: address: Authenticators 0x532eCF29b53f4c665cf31fA369a9362cd7699418
 1: string: De_Auth_Reason no deal power
 2: uint256: dAuthnums 2

(b) Check De-Authenticators status

Transactions 3 

0: string: Name IBM Servers
 1: uint256: Time_Stamp 1558524445
 2: string: Description network Server to head office
 3: string: Condition delivered before 30 days -4 units| must be match specs sheet : <https://www-07.ibm.com/servers/eserver/includes/content/pseries/downloads/factsfeatures.pdf>
 4: address: Creator 0x532eCF29b53f4c665cf31fA369a9362cd7699418
 5: address: Recipient 0xCfBaed77900707633b838f12DB7ce0f7713b66e8
 6: uint256: amount 0
 7: bool: Stop_Transaction true

(c) Check Transaction status

Figure 5.18: Status of the Transactions...

Add authenticators

Any Authenticator can add more Authenticators who are the stakeholders for further authentication process. please refer the figure : 5.19

Add Authenticator

TransactionID:

New_Authenticator_Address:

Purpose:



Figure 5.19: Add more Authenticators for farther Authentications

Remove stakeholder

Admin Can remove the stakeholder if he is no longer in the project. please refer figure: 5.20

Status and logs on ganache GUI

Status and logs on ganache GUI after the all process (please refer figures on 5.21.

Remove Stakeholder

Target Stakeholder:

Figure 5.20

ACCOUNTS	BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS	LOGS		
CURRENT BLOCK: 40	GAS PRICE: 2000000000	GAS LIMIT: 8721975	HARDENK: PETERSBURG	NETWORK ID: 5777	RPC SERVER: HTTP://127.0.0.1:8545	MINE STATUS: AUTOMINING	WORKSPACE: QUICKSTART
MEMORIC sibbling pizza symbol wolf lemon purchase birth holiday lecture ozone pill matrix						HID PATH 0/44-/00'/0'/0/account_index	
ADDRESS: 0xeC6BEca209Bc5ffbc4521EBDc43198c9918a988b	BALANCE: 1000.00 ETH	TX COUNT: 2	INDEX: 0				
ADDRESS: 0x42239eef76F90Fe768C0a8a8D95490242649A22	BALANCE: 1000.00 ETH	TX COUNT: 7	INDEX: 1				
ADDRESS: 0x532eCF29b53f4c665cf31fA369a9362cd7699418	BALANCE: 1000.00 ETH	TX COUNT: 16	INDEX: 2				
ADDRESS: 0x5cC8577Fb0e51c31D5f1EEA818F6f58eCfE570	BALANCE: 840.00 ETH	TX COUNT: 6	INDEX: 3				
ADDRESS: 0xEbD0B60D29c68e4915b8D2FC2d9d4AF16f6F3	BALANCE: 1000.00 ETH	TX COUNT: 1	INDEX: 4				
ADDRESS: 0xCfBaed77900707633b838f12DB7ce0f7713b66e8	BALANCE: 1160.00 ETH	TX COUNT: 7	INDEX: 5				
ADDRESS: 0xD6ab17322E35f38b6585e748B2c9213568c801E5	BALANCE: 1000.00 ETH	TX COUNT: 1	INDEX: 6				
ADDRESS: 0x44bE4319b53bc4B007FA432f77a2A25A09a7bE0	BALANCE: 1000.00 ETH	TX COUNT: 0	INDEX: 7				

ACCOUNTS	BLOCKS	TRANSACTIONS	CONTRACTS	EVENTS	LOGS		
CURRENT BLOCK: 40	GAS PRICE: 2000000000	GAS LIMIT: 8721975	HARDENK: PETERSBURG	NETWORK ID: 5777	RPC SERVER: HTTP://127.0.0.1:8545	MINE STATUS: AUTOMINING	WORKSPACE: QUICKSTART
BLOCK 40	MINED ON: 2019-05-22 17:52:12	GAS USED: 253747	<input type="button" value="TRANSACTION"/>				
BLOCK 39	MINED ON: 2019-05-22 17:52:09	GAS USED: 253945	<input type="button" value="TRANSACTION"/>				
BLOCK 38	MINED ON: 2019-05-22 17:51:58	GAS USED: 222719	<input type="button" value="TRANSACTION"/>				
BLOCK 37	MINED ON: 2019-05-22 17:44:47	GAS USED: 76186	<input type="button" value="TRANSACTION"/>				
BLOCK 36	MINED ON: 2019-05-22 17:43:11	GAS USED: 128494	<input type="button" value="TRANSACTION"/>				
BLOCK 35	MINED ON: 2019-05-22 17:41:11	GAS USED: 127375	<input type="button" value="TRANSACTION"/>				
BLOCK 34	MINED ON: 2019-05-22 17:39:19	GAS USED: 31225	<input type="button" value="TRANSACTION"/>				
BLOCK 33	MINED ON: 2019-05-22 17:39:15	GAS USED: 31225	<input type="button" value="TRANSACTION"/>				
BLOCK 32	MINED ON: 2019-05-22 17:39:10	GAS USED: 31225	<input type="button" value="TRANSACTION"/>				
BLOCK 31	MINED ON: 2019-05-22 17:37:43	GAS USED: 65542	<input type="button" value="TRANSACTION"/>				
BLOCK 30	MINED ON: 2019-05-22 17:34:30	GAS USED: 695251	<input type="button" value="TRANSACTION"/>				
BLOCK 29	MINED ON: 2019-05-22 17:20:54	GAS USED: 230377	<input type="button" value="TRANSACTION"/>				
TX HASH: 0x570c9b5a0ad6d64351ec4efb148b9bc7db81a6324f61adc7ef0d0294bc552e	<input type="button" value="CONTRACT CALL"/>						
FROM ADDRESS: 0x5cC8577Fb0e51c31D5f1EEA818F6f58eCfE570	TO CONTRACT ADDRESS: 0x0F29F37F553F60A0905F833bA342390c7F580a4	GAS USED: 253747	VALUE: 0				
TX HASH: 0x3d0bf858fed5c21aeb17707d2473b60dc5cbbcf8e2d8182961ae98e5e0e21e	<input type="button" value="CONTRACT CALL"/>						
FROM ADDRESS: 0x5cC8577Fb0e51c31D5f1EEA818F6f58eCfE570	TO CONTRACT ADDRESS: 0x0F29F37F553F60A0905F833bA342390c7F580a4	GAS USED: 253945	VALUE: 0				
TX HASH: 0x1f1a87c0aa3d32f290c962db712f7683b96ea99fc46966f26e3ad4e781ddd7a3	<input type="button" value="CONTRACT CALL"/>						
FROM ADDRESS: 0x5cC8577Fb0e51c31D5f1EEA818F6f58eCfE570	TO CONTRACT ADDRESS: 0x0F29F37F553F60A0905F833bA342390c7F580a4	GAS USED: 222719	VALUE: 0				
TX HASH: 0xb172944d6b3d5af13c862670c1128ec84ffc4d8567f8493d485da864452a06c	<input type="button" value="CONTRACT CALL"/>						
FROM ADDRESS: 0xCfBaed77900707633b838f12DB7ce0f7713b66e8	TO CONTRACT ADDRESS: 0x0F29F37F553F60A0905F833bA342390c7F580a4	GAS USED: 76186	VALUE: 0				
TX HASH: 0xf5e163b7354dca2e5b908c6ecf6963584daff71d0c408fe75575e2e6f227186	<input type="button" value="CONTRACT CALL"/>						
FROM ADDRESS: 0xD6ab17322E35f38b6585e748B2c9213568c801E5	TO CONTRACT ADDRESS: 0x0F29F37F553F60A0905F833bA342390c7F580a4	GAS USED: 128494	VALUE: 0				

Figure 5.21: Status of the Transactions...

5.1 Discussion

There are only prototype app and ethereum private network can have several limitations.

present financial regulations and procurement service of Sri Lankan government is very complex . when requesting wages from government office the project stakeholders must choose and filled forms according to the reason. as an example the form General-35 is used for requesting payment for procurement and services. how ever for the North western province they use General-26 form for the same purpose. also the D-Form is used if the officer sending a 3rd purpose to pickup his wage from his office.

Such that reason it is very difficult to implement such system for that. the researcher was follow only protect a general ledger information and financial Transactions with integrity,non-repudiation and the security.

The De-centralised system is build on top of the private blockchain. which have only one node that plant in one server. if some bogus user capable to change the blockchain, fraud can be detected but the original chain cannot be recovered. also the block chain server is hardly vulnerable to a DOS attack. due to having only one physical server with one node.

5.2 Future Work

- Prototype system can be implement as a real system with more functions and proper user interfaces.
- This system improve with a DAPP and it can host in cloud based.
- The system can be integrate with mobile app such a user friendly.
- this system can plant multi-node and multi-server private blockchain network with the help of privet-vpn it can serviced mobile users also.

Bibliography

- [1] G. C. Kessler. (2019, feb) An overview of cryptography. [Online]. Available: <https://www.garykessler.net/library/crypto.html>
- [2] D. M. Scott. (2018, oct) The essence of the blockchain. [Online]. Available: <https://medium.com/miracl/the-essence-of-the-blockchain-ac0c4f6dfeff>
- [3] R. Agrawal. (2018, may) Digital signature from blockchain context. [Online]. Available: <https://medium.com/@xragrawal/digital-signature-from-blockchain-context-cedcd563eee5>
- [4] L. Academy. (2019, feb) Blockchain basics. [Online]. Available: <https://lisk.io/academy/blockchain-basics>
- [5] F. Online. (2018, dec) 15 best accounting software systems for your business. [Online]. Available: <https://financesonline.com/15-best-accounting-software-systems-business/>
- [6] H. ANWAR. (2018, aug) Consensus algorithms: The root of the blockchain technology. [Online]. Available: <https://101blockchains.com/consensus-algorithms-blockchain/>
- [7] <https://coinmarketcap.com/>, 2019.
- [8] T. W. B. Group. (2019) The world bank in sri lanka. [Online]. Available: <https://www.worldbank.org/en/country/srilanka>
- [9] H. Ranaweera, “Real causes of project failures in urban development sector in sri lanka,” Architecture, Dept of Building Economics, University of Moratuwa, Motatuwa, 2016.
- [10] A. G. Department. (2016) The auditor general’s role and responsibilities. [Online]. Available: <http://www.auditorgeneral.gov.lk/web/index.php/en/roles-responsibilities>
- [11] M. B. C. DMCC. (2019) Annual audit and its importance. [Online]. Available: <https://mybusinessconsulting.ae/blog/annual-audit-and-its-importance/>
- [12] A. Obeyesekere. (2018) Transparency international - sri lanka. [Online]. Available: <https://www.transparency.org/country/LKA>
- [13] C. GOVERNMENT, *AN ORDINANCE TO PROVIDE A GENERAL PENAL CODE FOR CEYLON*. CEYLON GOVERNMENT PUBLICATION, 1885, ch. All.
- [14] P. O. T. D. S. R. O. S. LANKA, *AN ACT TO PROVIDE FOR THE PREVENTION AND PUNISHMENT OF BRIBERY AND TO MAKE CONSEQUENTIAL PROVISIONS RELATING TO THE OPERATION OF OTHER WRITTEN LAW*. THE GOVERNMENT PUBLICATIONS BUREAU,, 1954, ch. 26.

- [15] —, *FINANCIAL TRANSACTIONS REPORTING ACT, No. 6 OF 2006*, 1st ed. THE GOVERNMENT PUBLICATIONS BUREAU., mar 2006, ch. one.
- [16] D. G. of Public Finance, *FINANCIAL REGULATIONS OF THE GOVERNMENT OF THE DEMOCRATIC SOCIALIST REPUBLIC OF SRI LANKA*, 2nd ed. Sri Lanka: Department of Government Printing., 1992, ch. XIII.
- [17] N. P. COMMISSION, *PROCUREMENT MANUAL FOR GOODS, WORKS, SERVICES AND INFORMATION SYSTEMS – 2018*. Democratic Socialist Republic of Sri Lanka, 2018, ch. 5-11.
- [18] I. Lana Gates. (2019, feb) Why blockchain is important to business. [Online]. Available: insight.com/en_US/content-and-resources/2019/2202019-why-blockchain-is-important-to-business.html
- [19] Ethereum. (2018, feb) Ethereum project. [Online]. Available: <https://www.ethereum.org/>
- [20] P. Software. (2018, nov) Financial management erp software solutions - pronto xi erp systems — pronto software. [Online]. Available: <https://www.pronto.net/pronto-xi-erp/financials/>
- [21] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *www.bitcoin.org*, Tech. Rep., 2008.
- [22] T. C. Consultant. (2018, may) What is ripple (xrp)? explanation & advantages & disadvantages. [Online]. Available: <https://medium.com/swlh/what-is-ripple-xrp-337c91d8b636>
- [23] E. S. Z. W. C. P. Ahmed Kosba, Andrew Miller. (2016, may) Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. [Online]. Available: <https://www.computer.org/csdl/proceedings-article/sp/2016/0824a839/12OmNs59JYD>
- [24] J. K. Alex Chumbley, Karleigh Moore. (2016, apr) Merkle tree. [Online]. Available: <https://brilliant.org/wiki/merkle-tree/>
- [25] I. Watson. (2012, apr) How alan turing invented the computer age. [Online]. Available: <https://blogs.scientificamerican.com/guest-blog/how-alan-turing-invented-the-computer-age/>
- [26] guru99. (2018, jun) Blockchain tutorial for beginners: Learn blockchain technolog. [Online]. Available: <https://www.guru99.com/blockchain-tutorial.html>
- [27] Bitcon.com. (2019, feb) Bitcoin whitepaper: A beginner’s guide. [Online]. Available: <https://www.bitcoin.com/get-started/bitcoin-white-paper-beginner-guide>
- [28] Bitcoin.org. (2019, jan) How does bitcoin work? [Online]. Available: <https://bitcoin.org/en/how-it-works>
- [29] <https://etherconverter.online/>, 2019.
- [30] Ethereum. (2016, jan) Solidity. [Online]. Available: <https://solidity.readthedocs.io/en/v0.4.2/index.html>
- [31] <http://remix.ethereum.org/optimize=false&evmVersion=null&appVersion=0.7.7>, 2018.
- [32] B. A. Bouchefra. (2018, jun) Remix: Develop smart contracts for the ethereum blockchain. [Online]. Available: <https://www.sitepoint.com/remix-smart-contracts-ethereum-blockchain/>

- [33] H. Pokharna. (2016, jun) Develop dapps on ethereum. [Online]. Available: <https://medium.com/technologymadeeasy/develop-dapps-on-ethereum-tutorial-series-for-beginners-part-1-basic-terminology-866d2ce4cf34>

Appendix A

De-Centralized Secure Transparent Systems

```
pragma solidity >=0.4.22 <0.6.0;

contract Transparent_System {
    address public Admin;
    Transaction[] public Transactions;
    uint public numTransactions;
    mapping (address => uint) public Stakeholder_Id;
    Stakeholder[] public Stakeholders;

    constructor () public payable{
        Admin = msg.sender;
    }

    modifier Only_Admin {
        require(msg.sender == Admin);
        -;
    }

    modifier Only_Creater(uint TId) {
        require(msg.sender == Transactions[TId].Creator);
        -;
    }

    function Change_Admin(address New_Owner) Only_Admin public {
        Admin = New_Owner;
    }

    event Stakeholder_Changed(address Stakeholder, bool Is_Stakeholder
    );
    event TransactionAdded(string Tra_Name,string Tra_Condition,
    string Tra_Description, address Next_Recipient , address[]
    Authenticaters_List);

    struct Stakeholder{ //store and structure of the
        shareholders details
        address Stakeholder;
        string Name;
    }
```

```

        string Designation;
        uint Stakeholder_Since;
    }

    modifier Only_Stakeholders{
        for(uint J = 0;J < Stakeholders.length; J++){
            if(Stakeholders[J].Stakeholder==msg.sender){
                -;
                return;
            }
        }
        revert();
        -;
    }

    // adding a new shareholder to the pool of contact
    function Add_Stakeholder(address Target_Stakeholder, string
        Stakeholder_Name, string Stakeholder_Designation) Only_Admin
        public {

        uint id = Stakeholder_Id[Target_Stakeholder];
        if (id == 0) {
            Stakeholder_Id[Target_Stakeholder] = Stakeholders.
                length;
            id = Stakeholders.length++;
        }

        Stakeholders[id] = Stakeholder({
            Stakeholder: Target_Stakeholder,
            Stakeholder_Since: now,
            Name: Stakeholder_Name,
            Designation:Stakeholder_Designation      });

        emit Stakeholder_Changed(Target_Stakeholder, true);
    }

    //remove a shareholder from the list of shareholder pool
    function Remove_Stakeholder(address Target_Stakeholder) Only_Admin
        public {
        require(Stakeholder_Id[Target_Stakeholder] != 0);

        for (uint i = Stakeholder_Id[Target_Stakeholder]; i<
            Stakeholders.length-1; i++){
            Stakeholders[i] = Stakeholders[i+1];
        }
        delete Stakeholders[Stakeholders.length-1];
        Stakeholders.length--;
    }
} contract transaction_pro is ownrShip{

//store and structure of the transaction details
struct Transaction {
    string Name;
    string Description;
    string Condition;
    address Creator; //owner of transaction
    address Recipient; // next Authenticater of transaction
    uint amount;

```

```

        bool Stop_Transaction ;
        bytes32 TransactionHash;
        Authenticate[] Authenticaters;
        De_Authenticate[] De_Authenticaters;

    }

    struct Authenticate{
        address Authenticater;
        bool isApproved;
        string reson_To_Added;
        string Authentication_1st;
        string Authentication_2nd;
        string Authentication_3rd;
        address Added_By;
    }

    struct De_Authenticate{
        address De_Authenticater;
        string De_AuthReason;
        //string comments;
    }

    // create a new transaction for bigging of the process
    verification Transactions

    function newTransaction( string Tra_Name,string Tra_Condition,
        string Tra_Description, address Next_Recipient , address[]
        Authenticaters_List ) Only_Stakeholders public payable returns
        (uint TransactionID)
    {

        TransactionID = Transactions.length++;
        Transaction storage p = Transactions[TransactionID];
        p.Name = Tra_Name;
        p.Condition=Tra_Condition;
        p.Description = Tra_Description;
        p.Creater = msg.sender;
        p.Recipient = Next_Recipient;
        p.amount= msg.value;
        p.TransactionHash = keccak256( Tra_Name,Tra_Condition, msg
            .sender,Next_Recipient,msg.value,now);
        p.Stop_Transaction = false;

        for(uint256 i=0;i<Authenticaters_List.length;i++)
        {
            New_Authenticater( TransactionID,
                Authenticaters_List[i] ,msg.sender,"Initial
                Authenticaters");

        }

        New_Authenticater( TransactionID, msg.sender,msg.sender,"
            Initial Authenticaters");
        New_Authenticater( TransactionID, Next_Recipient,msg.
            sender,"Initial Authenticaters");
    }

```

```

        numTransactions = TransactionID+1;

        return TransactionID;
    }

    function Edit_Transaction( uint Transaction_ID, string Tra_Name,
        string Tra_Condition, string Tra_Description, address
        Next_Recipient , address[] Authenticaters_List ) Only_Creater(
        Transaction_ID) public payable
    {

        Transaction storage x = Transactions[Transaction_ID];
        x.Creater.transfer(Transactions[Transaction_ID].amount);
        x.Name = Tra_Name;
        x.Condition=Tra_Condition;
        x.Description = Tra_Description;
        x.Creater = msg.sender;
        x.Recipient = Next_Recipient;
        x.amount= msg.value;
        x.TransactionHash = keccak256( Tra_Name,Tra_Condition, msg
            .sender,Next_Recipient,msg.value,now);
        x.Stop_Transaction = false;

        for(uint256 i=0;i<Authenticaters_List.length;i++)
        {
            New_Authenticator( Transaction_ID,
                Authenticaters_List[i] ,msg.sender,"Initial
                Authenticaters");
        }
        New_Authenticator( Transaction_ID, msg.sender,msg.sender,"
            Initial Authenticaters");
        New_Authenticator( Transaction_ID, Next_Recipient,msg.
            sender,"Initial Authenticaters");
    }

    function getAuthMember(uint Traid,uint AuthId) Only_Stakeholders
    public view returns(address Authenticater , bool isApproved ,
    string Authentication_1st,string Authentication_2nd,string
    Authentication_3rd,address Added_By) {
        return(Transactions[Traid].Authenticaters[AuthId].
            Authenticater,Transactions[Traid].Authenticaters[
            AuthId].isApproved,Transactions[Traid].Authenticaters[
            AuthId].Authentication_1st,Transactions[Traid].
            Authenticaters[AuthId].Authentication_2nd,Transactions
            [Traid].Authenticaters[AuthId].Authentication_3rd ,
            Transactions[Traid].Authenticaters[AuthId].Added_By);
    }

    function approve(uint TracID,string _Comments) Only_Authenticator(
    TracID) public {
        bool isAllApproved=true;
        for(uint256 i=0;i<Transactions[TracID].Authenticaters.
            length;i++)
        {

            if(Transactions[TracID].Authenticaters[i].
                Authenticater == msg.sender){

```

```

        Transactions[TracID].Authenticaters[i].
            isApproved = true;
        if (Transactions[TracID].De_Authenticaters
            .length==0){
            Transactions[TracID].
                Authenticaters[i].
                    Authentication_1st = _Comments
                ;
        }
        if (Transactions[TracID].De_Authenticaters
            .length==1){
            Transactions[TracID].
                Authenticaters[i].
                    Authentication_2nd = _Comments
                ;
        }
        if (Transactions[TracID].De_Authenticaters
            .length==2){
            Transactions[TracID].
                Authenticaters[i].
                    Authentication_3rd = _Comments
                ;
        }
    }
    if(!Transactions[TracID].Authenticaters[i].
        isApproved)
        isAllApproved = false;
}
if(isAllApproved && !Transactions[TracID].Stop_Transaction
    )
{
    Transactions[TracID].Recipient.transfer(
        Transactions[TracID].amount);
}
}

function De_Authenticates(uint8 TracID,string De_Auth_Reason,
    string Comments) Only_Stakeholders public returns(uint) {

    for(uint256 i=0;i<Transactions[TracID].Authenticaters.
        length;i++)
    {
        Transactions[TracID].Authenticaters[i].isApproved
            = false; //ALL Authenticaters
            Approval set false
    }

    New_Authenticator( TracID, msg.sender,msg.sender,Comments)
        ;

    uint De_Authenticator_id = Transactions[TracID].
        De_Authenticaters.length++; //add De_Authenticator to
        de-authenticates list for ferther knolage
    De_Authenticate storage d =Transactions[TracID].
        De_Authenticaters[De_Authenticator_id];
    d.De_Authenticator = msg.sender;
    d.De_AuthReason = De_Auth_Reason;
}

```

```

        if (De_Authenticator_id>=2){
            Transactions[TracID].Stop_Transaction=true;
            Transactions[TracID].Creator.transfer(Transactions
                [TracID].amount);
            Transactions[TracID].amount =0;
        }
    }

function getDeAuthMember(uint Traid,uint AuthId) Only_Stakeholders
public view returns(address Authenticaters,string
De_Auth_Reason, uint dAuthnums) {
    return(Transactions[Traid].De_Authenticaters[AuthId].
        De_Authenticator,Transactions[Traid].De_Authenticaters
            [AuthId].De_AuthReason,Transactions[Traid].
                De_Authenticaters.length);
}

modifier Only_Authenticator(uint TId){
    uint x= Transactions[TId].Authenticaters.length;
    for(uint i = 0;i < x ; i++){
        if(Transactions[TId].Authenticaters[i].
            Authenticater==msg.sender){
            -;
            return;
        }
    }
    revert();
    -;
}

function Add_Authenticator(uint TransactionID, address
New_Authenticator_Address,string Purpose) Only_Authenticator(
TransactionID) public {
    New_Authenticator( TransactionID,
        New_Authenticator_Address,msg.sender,Purpose);
}

function New_Authenticator(uint TransactionID, address
New_Authenticator_Address, address Add_By,string
Reson_To_Added) private{

    uint Authenticater_id = Transactions[TransactionID].
        Authenticaters.length++;
    Authenticate storage a =Transactions[TransactionID].
        Authenticaters[Authenticator_id];
    a.Authenticator =New_Authenticator_Address;
    a.isApproved = false;
    a.Added_By=Add_By;
    a.reson_To_Added= Reson_To_Added;
}
}
}

```