



**UNIVERSITY OF
ALBERTA**

Masters of Science in Internetworking
**MINT 709 – Analysis of Blockchain Based Smart
Contracts**

Supervisor: - Juned Noonari

Author: Naveed Ahsan

2018-2020

Contents

1. Introduction to Blockchain	3
1.1 History	3
1.2 Bitcoin	4
1.3 Blockchain	5
1.4 Types of Blockchain.....	7
1.5 Consequences of Double Spending	11
1.6 Proof of Work.....	13
1.7 Ethereum: A Smart Contract Blockchain	15
1.8 Future of Blockchain	18
1.9 Smart Contract Platforms	22
2. Ethereum Platform	26
2.1 Proof of Stake	26
2.2 Relation between Ether and Gas	28
2.3 EVM (Ethereum Virtual Machine)	34
2.4 Ethereum Accounts.....	39
3. Smart Contracts	42
3.1 Solidity	42
3.2 How are Contracts Deployed	49
3.3 Demonstration of how Contract Works	63
3.4 Explanation of Code	78
3.5 Geth (Go Ethereum)	82
3.6 Geth Lab.....	83
3.7 Issues with Smart Contracts	96
3.8 The DAO Attack	98
3.9 Future of blockchain based Smart Contracts.....	99
4. Conclusion	102
5. Works Cited.....	110

1. Introduction to Blockchain

1.1 History: -

The concept of Blockchain was introduced to the world with the invention of bitcoin in 2008. Bitcoin started its practical implementation in 2009. It was built on decades of research on cryptography, digital cash, decentralization and distributed computing. It all started with the publication of paper by Satoshi Nakamoto 'Bitcoin: Peer to Peer Electronic Cash System'.

In this paper, the author proposes a peer to peer network using POW (proof of work) to record public transactions which requires no trust in contrast to traditional banking methods which relies on trust based models to prevent double spending. The proposed system is secure as long as the honest nodes in network control more CPU power than attacker nodes. [1]

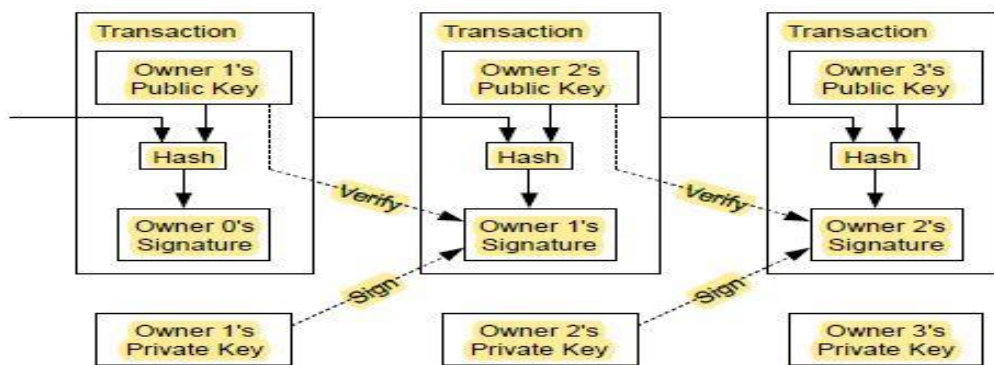


Figure 1.1 Proof of Work Architecture

[2]

A common solution is to introduce trusted third party that checks every transaction for double spending just like in banking system. In the Satoshi proposed architecture, each owner transfers the coin to the next owner by digitally signing a hash of previous transaction and the owner's public key. An electronic coin is the chain of transactions. A pay account can verify the transaction by public key and sign it with a private key.

The security method defined in this paper is Proof of work. POW is just one CPU represents one vote. To validate a block, the majority decision is needed. If the majority of CPU is

controlled by honest nodes, then the honest chain will grow and outpace the other chains. This will reduce the probability of attack. [3]

Here is a brief overview of important updates happening with Blockchain. In May 2010, first Bitcoin was purchased. In June 2014, a company named LHV started research on Blockchain. Ethereum project was launched in July 2014, which is a platform to develop applications. In September 2015, Major financial companies like CITI, NASDAQ committed to contribute to explore Blockchain technologies. They invested in a venture named chain.com. By 2016, 40 major financial institutions have invested in Blockchain.

Blockchain evolved from peer to peer digital payment systems to be used for developing wide range of decentralized applications. There is no central authority in Blockchain. Blockchain is a decentralized ledger maintained by a group of independent users. A ledger is a physical book that holds records of transactions. It records all the transactions and then executes them over the nodes. This information will be sent from node to node. Some of the nodes in between will verify the legibility of information before passing it ahead. Those nodes will be rewarded and the information will be made public or private depending upon the type of Blockchain.

1.2 Bitcoin: -

Bitcoin is the first application of Blockchain technology. The idea to create digital cash goes as far back as 1980's. David Chaum presented a scheme that uses blind signatures to build digital currency. In this scheme, bank would keep a serial number and issue the same to user. The problem with this scheme was that bank was the central authority. Later on David Chaum presented another idea that along with serial number, bank will give some private identification data to craft the message. In this scheme, double spending became a problem.

Adam Back's hashcash introduced the idea of solving computational puzzles. In 1998, Nick Szabo introduced the idea of proof of work. But the system proposed lacked basic consensus mechanism between nodes. Hal Finney introduced Reusable Proof of Work (RPOW). Major drawback for this system was that central authority keeps track of all POW tokens. [4]

All the ideas presented by David Chaum, hashcash, Hal Finney lead to the creation of bitcoin. All the technologies like BitGold, B Money and cryptographic time stamping are combined in

the creation of the world's first decentralized digital currency. Bitcoin with a capital B is the protocol while b represents currency. [5]

In the past, digital currencies had a problem of double spending. Bitcoin tackled this problem by introducing the verification of public ledger, it means bookkeeping all transactions of bitcoin since its invention. Since this is a P2P decentralized network, so all the transactions can be traced back to its creation. As per Kings, Williams and Yanosfky '*the first thing that bitcoin does to secure ledger is decentralizing it. There is no master spreadsheet anywhere*'. There is a hash algorithm validating the transactions because it's compulsory to sign the transactions. After this, block is added and distributed to network. So all the miners will know the existence of block and whether to attest it or not. [6]

There is some criticism associated with bitcoin as well. Some critics believe that transaction costs are not clear cut, which leads many users to believe that transactions are free. But in reality miners are earning 3.5 percent of the transaction. Cryptocurrencies are generally more criticized on money laundering grounds. According to some critics, bitcoin has made some kind of 'criminal paradise' because it is not immune to theft. There are some incidents reported in which bitcoin has been stolen. A drug selling company has reported that someone has stolen their bitcoin worth millions of dollars. Bitcoin is also providing a way for tax evaders. Since it's not a regular currency, governments are having hard time finding real tax evaders. One other disadvantage is that there is a limited amount of bitcoin in the world. It is analogous to amount of gold in world, it's finite.

Despite not being the only cryptocurrency, bitcoin is the most famous one among all of its competitors. One of the many reasons for its popularity is that bitcoin is the only currency on this big scale. Major companies like Dell have started accepting Bitcoins. [7]

1.3 Blockchain: -

Definition of Blockchain and how Blockchain works are already briefly described above. Here are some of the prominent features of Blockchain that make it more beneficial as compared to other technologies.

- Blockchain stores information in batches called blocks. Blocks are linked together in chronological fashion to form a line. If we make a change in one block, it will be stored in a new block that will be added at start of line.
- Blockchain eliminated the need for middle man. Blockchain provides trusted peer to peer interaction without dependency on any intermediary party. Blockchain can revolutionize how we access, verify and transact with each other.
- Blockchain creates trust in data. Cryptography is a mathematical function used in encryption and decryption process. First cryptographic puzzle has to be solved. Computer that solves the puzzle shares the solution to all computers in network called POW. Network then verified proof of work. If correct, new block will be added to the chain. Because network does the trust building, so we have the opportunity to interact with data in real time.
- Blockchain is an open source technology. It's not limited to some specific organization or company. All operations in Blockchain are open to the community to perform.
- Blockchain is a really secure technology. Due to the presence of ledger and POW, it's nearly impossible to tamper data or add new data without verification from the network.
- It's an automated technology over the software. People don't need software companies to help them with transactions. There is no single owner in Blockchain. It's a trusted automated model in which customers can interact by themselves without any supervision.
- It works in a distributed manner. All the records are stored on all the nodes across network. So even if one node goes down due to power outage or other issues, other nodes still have sufficient information to perform the transaction.
- Blockchain is a really flexible technology. It can be programmed in any way possible using basic programming concepts and programming semantics.
- Blockchain is a cheap technology. We don't need any type of financial power to perform or interact in Blockchain. You can do transactions in Blockchain if you have a simple personal computer and operating system inside it. As there is no third party in Blockchain, it can massively save the amount of cost given to parties in order to perform transaction.

Here's an image depicting a traditional centralized system above while a Blockchain based decentralized system below.

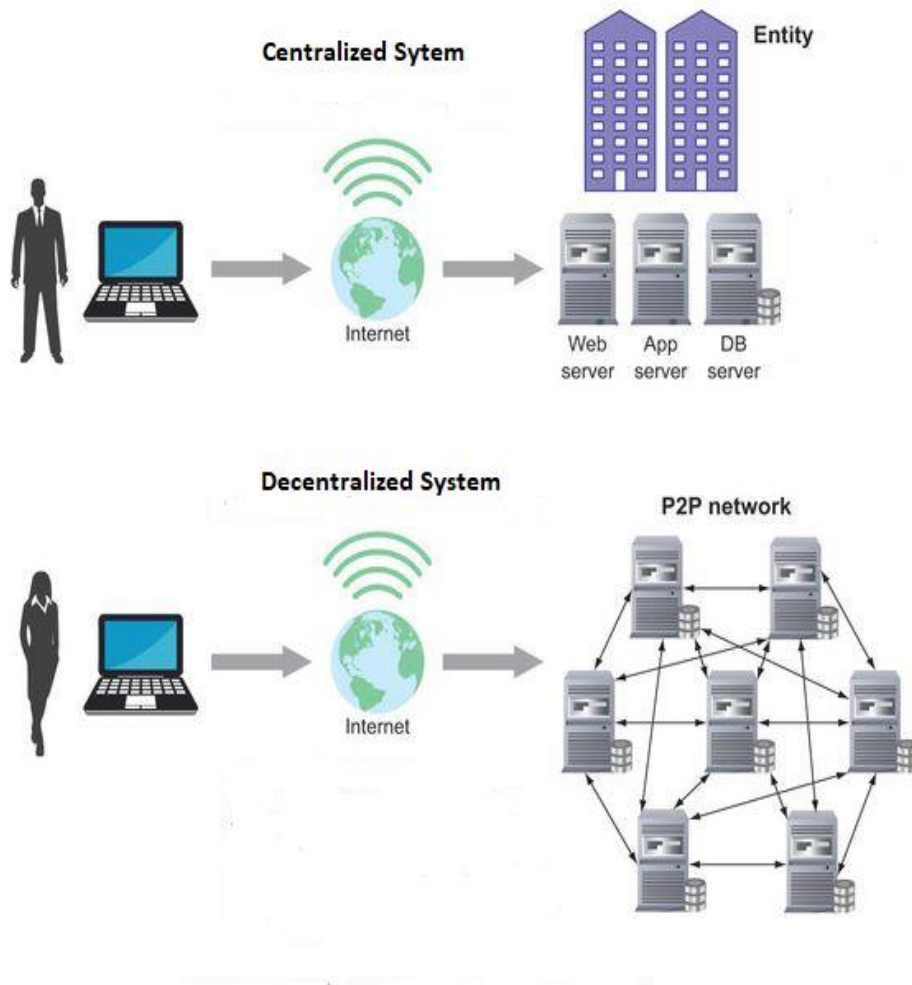


Figure 1.2 Comparison of Centralized and Decentralized System

[27]

1.4 Types of Blockchain: -

There are different types of Blockchain available to users around the world. Most prominent ones are: -

Public Blockchain: No single person or company owns this Blockchain. Anyone around the world can participate. Ledger in public Blockchain is visible to everyone on the internet. User details are visible to everyone and anyone can verify or add information in Blockchain. Public Blockchain are also called permissionless Blockchain. Major examples are Bitcoin, Ethereum, Dash.

In public blockchains, anyone in the world can take part in transactions already going on and anyone can leave the network without others consent. The openness of public Blockchain have led to problems like bad performance, and people have used these features as a bad influence to manipulate Blockchain. People exhibit Byzantine behaviour that is a wastage of mathematical and technical resources. PBFT is explained briefly.

Byzantine General problem is used in computing science for situations where consensus is necessary in system regardless of malicious nodes. Byzantine Fault tolerance optimizes the aspects of byzantine general problem. BFT is used in distributed computer networks to reach consensus with malicious nodes present. BFT can only work if the number of malicious nodes are less than honest nodes. In PBFT, all nodes in system communicate with each other with a goal to reach consensus. BFT is highly being used in Blockchain technologies. Hyperledger Fabric uses PBFT a permissioned version to reach consensus. Since permissioned chains use small consensus groups, PBFT is perfect to be utilized there. [31]

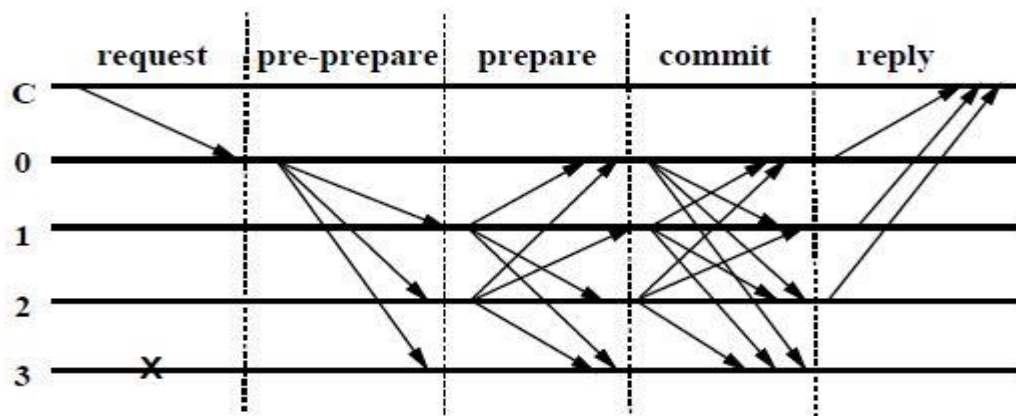


Table 1 Depiction of how PBFT works

The above diagram shows five nodes or processes. This is how practical byzantine fault tolerance works. C represents client, 0 can be named Clive, 1 is Sarah, 2 is Michael and 3 is Laura.

Client sends request to Clive, this is process 0, the initial request. During the same time Laura fails. Then Clive sends a pre-prepare messages to everyone. Everyone except Laura responds back with prepare message. After acknowledging everyone presence, all sends

commit messages. After hearing sufficient amount of commits, we respond directly to client. [32]

There are many advantages of public Blockchain, but with advantages comes disadvantages too. Public Blockchain is notably slow. Ethereum got some upgrades in terms of speed from bitcoin but still it's no comparison to a centralized system. The main reason for the slowness is the consensus algorithm needed to verify every transaction. And there are restrictions on how many transactions can fit in one block and the time needed to process one block. With the slow processing power, comes the scalability issues. Current public block chain cannot compete with the rising number of transactions. With more participants and usage, it will clog the network. Energy consumption is also a major issue related with public blockchains. But with new consensus algorithms it has been tackled to some extent.

Main disadvantage of public Blockchain is its public nature itself. There is no privacy for transactions or participants joining the network. This makes them unsuitable for enterprise level domains.

Private Blockchain: Not everyone can join this Blockchain. It's only restricted to some specific users. This Blockchain allows only specific users to add and verify blocks in Blockchain but anyone on internet can verify them. All permissions are kept within the organization that controls the Blockchain. Private Blockchain are also called permissioned Blockchain. Many big companies like IBM, Huawei, AWS, SAP and Baidu have recognized the importance of Blockchain in fields other than cryptocurrencies and have their research focused on developing products on Blockchain. Examples are Ripple, Hyper ledger.

There are certain entities like this kind of Blockchain that allows users to be a part of chain. Since in this kind of Blockchain, real-world identities of people are generally known so there is no need for expensive consensus algorithms like POW. A specific type of protocol is used to avoid any kind of branching. Security, performance and scalability are handled much better than a public Blockchain.

IBM have developed a private Blockchain named Fabric. Fabric has been used as a project in Hyperledger consortium by IBM. Hyperledger Fabric is now being used in five different industries under the umbrella of IBM. Food source tracking, airline industry,

enterprise operations management, a trail of cambio coffee and insurance compliance data are the five areas of IBM's interest. Currently there are many vendors supporting hyperledger fabric and providing Blockchain as a service to customers, for example, IBM Blockchain platform (IBP), Oracle Blockchain platform, Amazon Managed Blockchain, Microsoft Azure Blockchain Workbench, Alibaba Cloud BaaS, Baidu Blockchain Engine, Huawei Blockchain Cloud Service and SAP Cloud Platform Blockchain.

Oracle has incorporated hyperledger fabric in its own BaaS which has provided better results and efficiency. Oracle Blockchain platform used SQL. But the database enhancements done by oracle are not open sourced. [28]

Huawei has proposed an end to end Blockchain architecture that integrates network, TEE (devices and chips) and Blockchain platform. TEE is trusted execution environment that can be used to optimize operational procedures of Blockchain. Huawei believes that with the increasing use of Blockchain and technologies like Hyperledger, there might be a need to reconsider the P2P network being used. Huawei has suggested edge computing could be incorporated into Blockchain that could elevate processing and storage problems being faced from large number of nodes. Huawei's research of including more network equipment into Blockchain can be further evaluated. The below image shows Huawei approach to Blockchain.

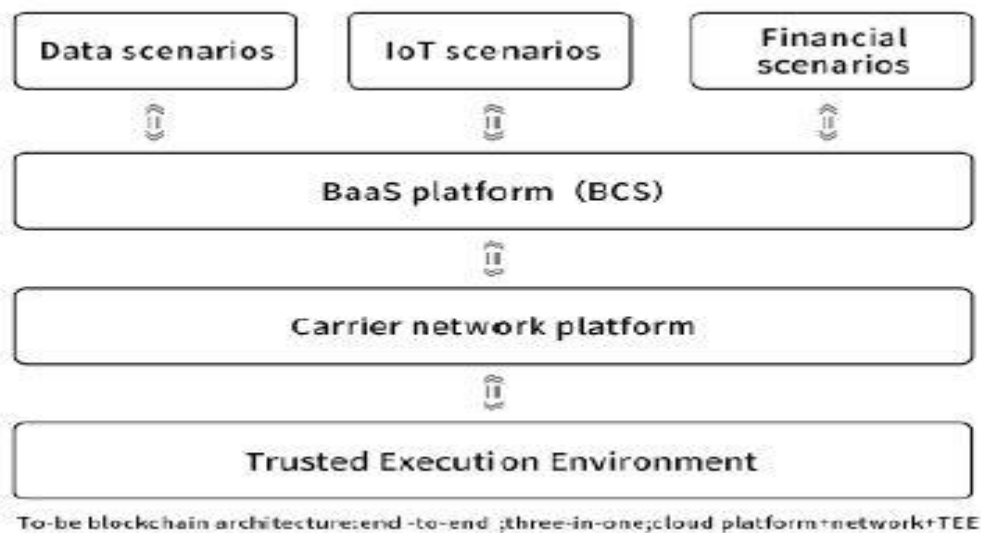


Figure 1.3 Huawei's Blockchain Hierarchy

Huawei is working on its own Blockchain service called BCS. BCS is designed on Huawei cloud which is an open source technology. Blockchain service of Huawei have many advantages on other competitor chains. For example, BCS is more user friendly due to its automation capabilities, it's an open source technology built on open source materials so that everyone can make use of it, BCS comes with appropriate tools for integration with cloud and since BCS is built on Huawei cloud, it inherits all the capabilities of user management, cyber security and permissions. Most definite advantage of BCS is improved security in Blockchain. Security features like consortium are controllable, transactions are traceable, undisputable, visible and auditable, transactions are performed anonymously and use distributed ledger technology to provide encrypted transactions. [30]

Consortium Blockchain: This type of Blockchain is only controlled by the consortium of members. Normally, major companies that build this Blockchain are the ones controlling the permissions, adding and verification of blocks in Blockchain. Changes are made only for specific purposes. Examples are hyper ledger 2.0. [8]

Semi-private Blockchain: Part of Blockchain is public and part is private. Private part is controlled by specific members while public part is open to everyone.

1.5 Consequences of Double Spending: -

In a real world, when you buy something with a note or a coin, you can't double spent it. It means you cannot buy two different things by the same coin. However, that's not the case with cryptocurrencies. In cryptocurrency you can use the same bitcoin or ether for two transactions. That is called double spending. For example, if a person A sends ether to Person B. He can send the same ether to person C as well. There are many ways double spending can happen. Most common ones are 51% attack, race attack, Finney attack, vector 76 attack and alternative transaction attack.

In 51% attack, if a miner has more computing power than whole of Blockchain. Then he will overpower the Blockchain and make his own Blockchain the honest one. And then he can create more blocks faster than usual with more hashing power. This attack mainly depends on the computing power of dishonest miner. Because difficulty of puzzle needed

to solve and the principle that longest chain wins in Blockchain, makes it almost impossible to falsify the Blockchain.

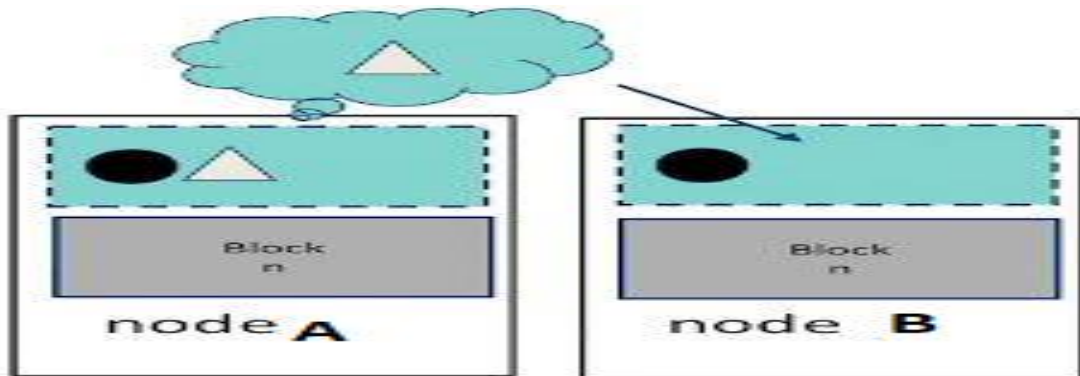


Figure 1.4 Representation of Two Nodes

[15]

For example, there are nodes in world at different locations. One is in Edmonton Canada and other is in Lahore Pakistan. Both these nodes are part of the same Blockchain. What happens is that both these nodes are listening for transactions. These nodes will compete with each other in building enough transactions to be a full block. If a person says, he is going to send money from account to another. So node A listens to this transaction. It will broadcast the transaction to node B. So then both nodes will enter this transaction in Blockchain.

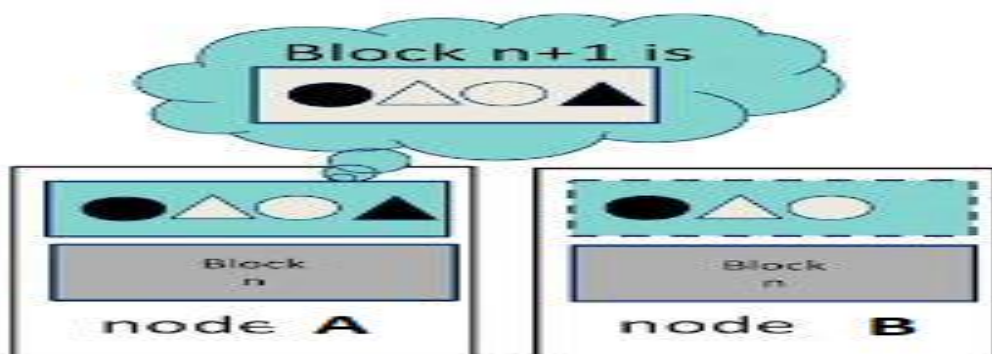


Figure 1.5 Demonstration of two nodes agreeing to add another node

So block n has built completely. Now nodes are competing to build block n+1. Each node is working on a complex mathematical problem, whichever nodes does it first, wins a prize and broadcast the results to all nodes. All other nodes confirm the result and add the

new block to the chain. Since every node is trying to build block by listening to transactions. Even if the transactions are in different order, nodes will correct that after confirming the mathematical result. That's called proof of work. [16]

So Ethereum Blockchain prevents double spending by publically announcing transactions. Even if a node didn't learn about a transaction at first time due to unknown reasons, it will learn this time from other node broadcasting the transaction. Secondly each node keeps track of a chain of block of transactions.

1.6 Proof of Work: -

Proof of work algorithm is the first created algorithm in Blockchain. This Consensus based algorithm was developed to be used in bitcoin. The general idea behind this algorithm is that every node tries to solve a complex mathematical problem based on cryptographic hashing. As already mentioned above, each node tries to build a new block and append it to Blockchain. Whichever mining node solves the puzzle first, gets a reward. And the result of puzzle is broadcasted to all other nodes, then they verify that result. If that result is accepted by every node, then the miner node gets a reward. This is how Blockchain maintains a distributed ledger without keeping a central authority.

Every block in Blockchain is linked to each other by previous block hash value. Each block has a unique hash value. POW algorithm uses a hash value. So when a new block is appended, every field in the header of block is filled except the Nonce value. Nonce is 32-bit fixed length data. The nonce value must be very long and difficult to crack because if it is easy, everyone will try to add new blocks in chain so that they can win reward. The difficult target is to find a hash value with the required number of zeros in the start of the nonce so that it can be part of block header hash. So whenever a new block is ready to be appended, nonce value has to be found and the only way to find that is to do repeated hashing. [18]

To make it understand more easily, we will look at a modified example. We will assume that block header equals to sentence and that sentence is followed by a number that is nonce value in our case. For simplicity we will set a hash function that starts with two zeros. The sentence will be 'I am a good boy' and difficulty value is 2. So after 10

attempts we will be able to crack that hash value. More recently for bitcoin, we need to match the hash function up to 72 zeros.

I am a good boy 1 = 2544abd889e45...

I am a good boy 2 = ef56738277748...

I am a good boy 3 = 234aaab555666...

I am a good boy 4 = de6873e777744...

I am a good boy 5 = acd56333134dd...

I am a good boy 6 = 89898989dccc...

I am a good boy 7 = abcde55757575...

I am a good boy 8 = 5788abcdef444...

I am a good boy 9 = 78345aad78934...

I am a good boy 10 = 2bbc87637935...

Sometimes it happens that after the puzzle is solved and the block is ready to be appended. Nonce value is found and the result of POW is broadcasted. Some other node solves the puzzle and broadcast his solution. So it's not necessary that every node have same ledger. Some nodes have same ledger but in different sequence. This is called forking problem. Every node continues with the block they perceived. So after sometime, chain with the longer block wins. Because it has more mining power so the probability of extending the chain further is even more. So whenever a fork occurs, we continue with the longest chain. [18]

Hashing is the process of transforming any input data into a random string of characters. It is almost impossible to derive input data from output hash value. Even a slightest change in input value can create a lot of change in hash value. No matter what the size of input value is, length of hash value always remains the same. It is not mathematically feasible to differentiate two input values on the basis of output hash value. POW algorithm uses SHA-256 hashing. SHA256 always generate an output of 256 characters. SHA256 is used in other parts of Blockchain as well. [17]

1.7 Ethereum: A Smart Contract Blockchain: -

Ethereum is an open Blockchain platform that lets anyone built and use decentralized applications that runs on Blockchain.

A smart Contract is just a computer program in Blockchain. This program is immutable and distributable. Smart contracts are just like normal contracts except digital. Smart contracts are very much like the object-oriented classes. One smart contract can call other smart contracts and use objects of another class as well. Smart contracts are codes that are deployed and executed on the Ethereum virtual environment.

Below figure shows how a smart contract works. Everything is done virtually on a Blockchain and digital currency is used. Blockchain makes sure that the contract is viable and there is no fraud in this exchange.

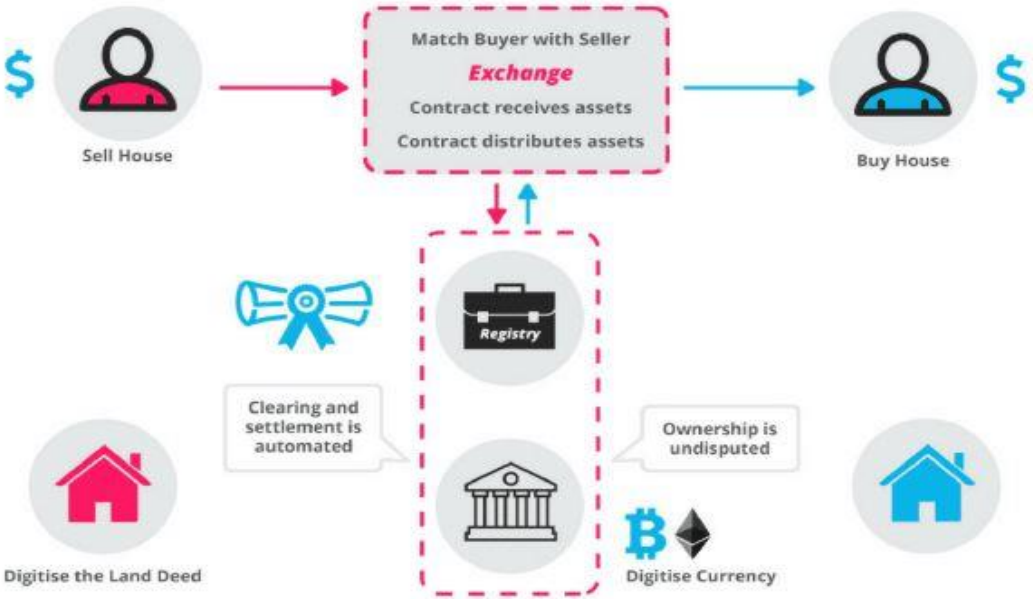


Figure 1.6 Demonstration of How smart contract works

[40]

Ethereum is not the only Blockchain you can deploy smart contracts into. But the features that make it distinctive are that it is open source, public Blockchain. Anyone can check the logs and every transaction is visible to outside world. Main purpose of Ethereum is to help developers create applications on a decentralized network. We can perform Bitcoin like transactions or more complex functions.

Some of the critics say the invention of a decentralized network it has lead the way to web 3.0. Web 1.0 uses static websites like uofa.edu. Then comes Web 2.0 that used dynamic websites like Facebook, Twitter, Amazon. In both first versions of Web, you should have a webserver whether the server is centralized or decentralized around the world. But in Web 3.0 there is no need for a server. So when you are working on the decentralized application, you are interacting directly with Blockchain on the other side. Each piece of information on Blockchain is verified by other nodes and your identity is saved and protected by Blockchain.

Every time you run a command or write a program, it runs on every node in Ethereum. Storing the information works the same way. So generally doing all this in all nodes around world is costly. To avoid this waste of resources, there is an interplanetary file system or IPFS. For example, if you just want to store a part of any legal document, so you will hash in an exclusive way, store the hash on Blockchain and full document on IPFS. [9]

If two decentralized apps want to communicate with each other, they can use the P2P protocol in Ethereum called whisper. There are a number of reasons why Dapps need to communicate without going through Blockchain. Some of the prominent reasons are privacy, cost of on-chain transactions or if Dapps needs to collaborate on the transaction without going through Blockchain. Whisper can be implemented in Geth client. It is disabled by default and needs to be programmed by Dapps for communication.

Whisper protocol is built on a darkness principle. This means that all the messages sent between sender and receiver are encrypted, so the information in the message cannot be gained by packet analysis. Whisper protocol is built on the top of RLPx transport protocol that is internally used by Ethereum for communication between nodes. There can be spam in a whisper as well, which can be avoided by executing Proof of work algorithm. [10]

Smart contract contains the specific terms that the buyer and seller have agreed upon. The same contract is distributed among all the nodes in network. This means that all the participants know and agree to the contract, then this smart contract will execute. If the same smart contract runs on each node, the result will be the same.

Main components of smart contracts are set of executable variables and functions.

Whenever a smart contract is executed, the values of these variables and functions are

changed. Smart contracts in Ethereum are written in solidity language. Smart contracts can send each other messages. Whisper protocol is used for sending messages. Smart Contracts are mainly deployed and developed in three types of Blockchain, Ethereum, bitcoin and NXT. [11]

Researchers have decided on a blockbench having different layers to compare all types of Blockchain around the world. This blockbench model has four layers. Every Blockchain from bitcoin to Ethereum can be incorporated into it. The consensus layer decides on whether to append a block or not and what protocol to follow for it. Ethereum uses a proof of stake POS algorithm as a consensus algorithm. POS selects a node that can decide whether to append a block or not basis on its investment.

The data layer contains the structure, content and operations of Blockchain. Ethereum contains special accounts called smart contracts, which in turn contain private states. In Ethereum balances are a part of states and they are updated with ongoing transactions. When an account receives a transaction, the contract is invoked which then checks the terms specified in the transaction. The execution layer determines the environment in which Blockchain operations run. Ethereum has its own virtual machine called EVM Ethereum virtual machine. Solidity is used for programming the smart contracts. Every code that runs in EVM costs a certain amount of gas and that gas is to be returned if the transaction is not correct. EVM is optimized in Ethereum by itself.

The last layer called the application layer, contains the classes of Blockchain applications. Ether is the currency for Ethereum Blockchain and every smart contract that runs in it has to use this currency. Decentralized autonomous organization DAO is the most famous application that runs in Ethereum for crown funding, investment and other decentralized activities.

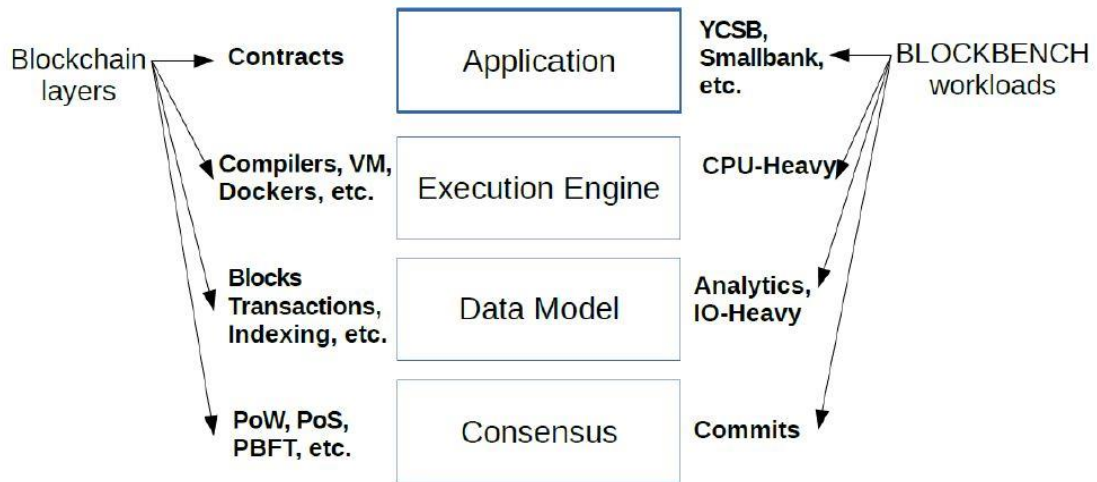


Figure 1.7 Blockchain Layers and BlockBench Workload

[29]

1.8 Future of Blockchain: -

The future of Blockchain is very promising and very futuristic. Web 3.0 has already become available alongside web 2.0. For example, we have Brave as a browser, IPFS for storage, Expertly for audio and video calls, status for messaging. All these applications are already in the market and numerous more are coming in the future. A pie chart is drawn below to show Blockchain-based applications in different fields of computing.

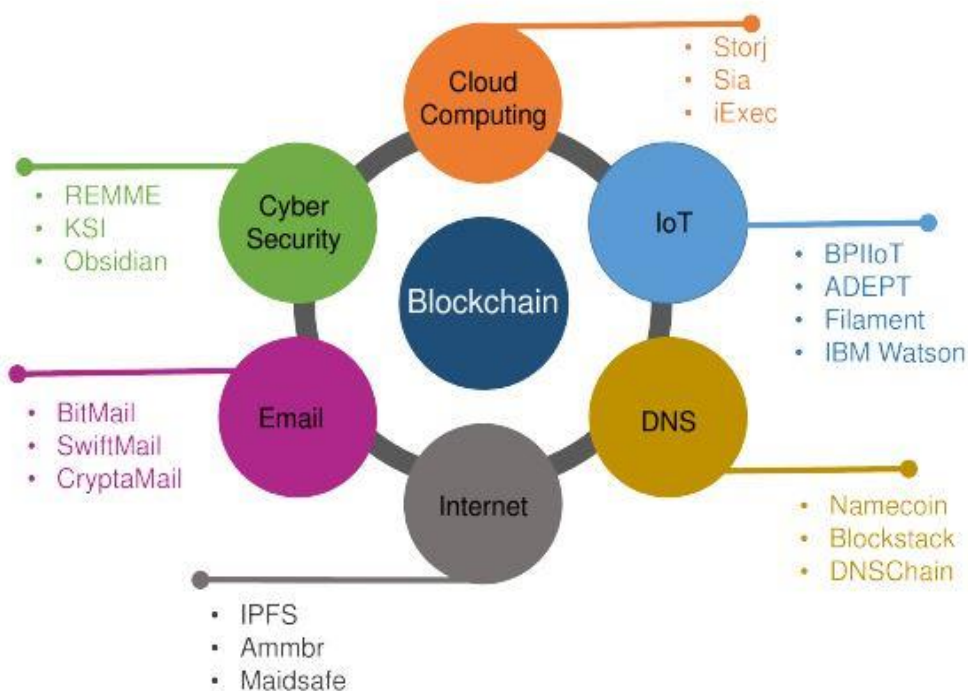


Figure 1.8 Pie Chart of Blockchain Applications

[21]

Blockchain in future is going to make a major impact on financial industry. Using Blockchain in financial transaction has multiple benefits like reduction in fraud, increased efficiency, Improved customer experience. Financial institutions use a centralized system for transactions and database management. Centralized systems are more vulnerable to attacks. And more importantly it's a single point of failure. Blockchain provides a decentralized system with a cryptographic mechanism which in turn is more secure. Providing a single ledger that is synchronized with the network all over the world eliminates the need for reconciliations. Some experts have predicted that with the use of Blockchain in industry, customer experience will be 25 percent improved due to faster processing. [20]

Along with the growth of Blockchain, research has been going along the way to make Blockchain safer and more convenient to use. Centralization issues, limitations in cryptographic functions, consensus algorithm, scalability are the major areas of research. With the growth of bitcoin mining farms, there is a growing concern to make it decentralized again. More research in the area of smart contracts has been done to make it

more secure and safe. Proof of stake algorithm has already been developed in order to overcome the issues of power consumption with proof of work algorithm. Issues related to scalability has been addressed by increasing the capacity of Blockchain rather than using side channels.

Aside from the financial industry, Blockchain can be used in many different fields in future. Content distribution networks are implemented to improve quality of service. Users in different regions can access data from nearby distribution networks rather than going to originating server. Netflix, YouTube and Google have CDN's all over the world. Blockchain technology can improve CDN's in many ways. It can provide better ways of content control, more trusted and intelligent ways of delivering content. Users can independently verify a record and its origin without going to a central authority. Famous example of Blockchain based CDN is DECENT. Blockchain based CDN maintains the reputation of content creator by providing a secured way of payment from client and author nodes. [22]

Another use of Blockchain can be in distributed cloud storage. Cloud storage has become really popular in recent days due to the availability of services like google drive, dropbox. Anyone can purchase non-volatile storage from these companies and use that storage for personal or commercial use. But these services follow the client-server model of TCP/IP. Almost all the cloud storage nowadays use the traditional TCP/IP model for communication between a user and their servers. The main problem in this approach is the hackers, thieves and censorship agencies can tamper or copy the data creating duplicate servers through illegal and technological means. These problems are caused due to centralization of cloud storage systems. And this can be potentially solved by decentralization and a trusted agreement between host and service provider. By implementing Blockchain in cloud storage systems, properties like decentralization, immunity, anonymity and trusted execution of data transfer and storage can be adopted. Additional features like scalability, high redundancy and security also come with the implementation of Blockchain.

Storj, Sia and Filecoin uses Blockchain as a means of distributed cloud storage. Storaj is a point to point cloud storage platform, which enables users to share storage as per their own will with third party providers. Users can earn cryptocurrency by selling their unused data storage. Sia is another application of Blockchain based on distributed cloud storage. Sia

uses service level agreements between user and provider mentioned in smart contract. It's an open source platform that splits data into pieces, encrypts that pieces and spreads it across the network. This approach is way more reliable and secure than centralized storage systems. Filecoin is based on IPFS (inter planetary file system). Miners in Filecoin holds the storage space. Storage space held is determined by the mining capability of miners. Each transaction is done in currency called Filecoin. [23]

A paper published by changyu dong, amjad aldweesh, yilei wang on smart counter collusion contracts for verifiable cloud computing states to make clients enable to analyse a collusion between two different clouds be making them perform same computing task. Users will use smart contracts to know about distrust, tension among different clouds in fact avoiding collusion and cheating.

Some researchers are working on integrating Blockchain in online social networks and cybersecurity. Cybercrime has become a major issue in Blockchain as well. A DDoS attack on Blockchain rendering floods of fake transactions. These transactions are cheaply valued, so attackers take advantage of that. When it comes to smart contracts, there is much larger area where attacker can work. A small error in smart contract can result in major losses. If one smart contract malfunctions a domino effect is created and all other smart contracts lose their credibility too. A similar attack happened on Decentralized Autonomous Organization (DAO) on a Ethereum based Blockchain resulting in loss of 60 million ethers. REMME and Obsidian are Blockchain based technologies to prevent cybercrimes. REMME is password authentication system for safeguarding confidential information and disregarding the need of password memorization. Obsidian is used to secure messages without any dependence on centralized platforms. The metadata about the ongoing communication is stored in distributed ledgers and cannot be collected simultaneously. [24]

In recent times, people have largely engaged with online social networks. These OSN's (Online Social Networks) let people communicate on a free medium. A large of amount of personal and confidential information is shared on these networks. Hence keeping these networks secured is a major necessity of time. An incident like Cambridge Analytica claimed the information of more than 50 million users during US presidential elections. A Blockchain based online social network called steem enables users to earn rewards on the basis of their participation. Users decide which content to publish and what not to. This

encourages an honest participation of community. Blockchain based ‘reputation system’ keeps records of users based on their transaction history. [25]

In future, Blockchain can also be used to create a decentralized internet. Internet used today is mainly centralized. It has revolutionized many services like e-commerce, health, online social networks and digital financial systems. But with this much innovation, cybersecurity has become a major issue with centralized internet. Decentralizing internet means giving more authority and rights to users and it can also be used to bridge the digital gap, make internet services reachable to places where it hasn’t been before.

Domain Naming System (DNS) is a prime example of today’s internet evolution. It’s a namespace server used to resolve names into IP addresses and vice versa. But this centralized approach has problems like single point of failure and makes it easier for attackers to use their malicious activities. Blockchain can provide the naming system based on append-only ledgers, therefore, guarantee the availability, uniqueness and integrity. One such example of Blockchain based naming system is called Name coin. Name Coin being based on top of Blockchain requires POW by longest chain of honest peers to change any name entries already in ledger. Name coin is designed to provide improved security, agility, decentralization and confidentiality.

The prime concern of some researchers is routing in decentralized internet. There is a need for routing in the future if we want to connect all the Blockchain networks to create a major decentralized chain. If we connect two different chains, they will have to verify their records by reaching a consensus on a mechanism like POW. Interoperability between different chains comes from the concept of lightweight clients. Such clients verify the data by downloading the exact needed data rather than downloading the whole Blockchain. Blocknet uses the similar principle for routing between different chains with two main components called Xbridge and Xrouter. Xrouter is used to enable the communication functionality between chains and Xbridge is used for enabling the swaps of tokens between chains. In essence, the whole scenario is closely related to BGP in traditional routing where different autonomous systems communicate with each other. [26]

1.9 Smart Contract Platforms: -

Ethereum is not the only Blockchain in which we can implement smart contracts. There are few notable blockchains available other than Ethereum to deploy smart contracts. Most famous are EOS, rsk, stellar, cardano, Neo, Lisk, Stratis and hyper ledger fabric.

EOS is one of the most famous contender to Ethereum. It was developed by Dan Larmer and Brendan Blumer. Dan Larmer is the one who invented concept of POW and decentralized autonomous organizations as well. Block One is the organization where EOS was founded. EOS works on EOSIO software. Main motivation behind invention of EOS was slow operation of Ethereum blockchain. Ethereum can only process 15 transactions in one second. Keeping in view rapid growth in blockchain, Ethereum is not a good alternative because of its speed. EOS is developed to support industrial size applications. EOS works on decentralized operating systems rather than decentralized computer. Users in EOS own stake as per their resources. So if someone owns 10th part of stake in EOS, this means he has 10th part of computational power of EOS. Consensus mechanism used in EOS is delegated proof of stake. In this type of consensus mechanism, anyone who owns tokens in blockchain integrated in EOS software can approve appending of block based on continuous voting system. EOS uses Web assembly to code smart contracts. WASM web assembly will let coders develop smart contracts in their own language of choosing and compile bytecode onto supported browser. [73]

Cardano is one of the newest contenders in smart contract blockchains. Cardano provides layered architecture for smart contracts. Because of its layered architecture, cardano provides system the flexibility to be easily maintained and allows the updates by soft fork. This blockchain uses a new POS algorithm called ouroboros. Cryptocurrency used in Cardano is Ada. Ouroboros is designed by chief scientist Aggelos Kiayias. This is the first consensus algorithm approved by peer review in major blockchain conference and is proven mathematically to be secure. Main idea behind Cardano invention is to regulate financial services by providing open access to users. [74]

Programming languages used in Cardano are Haskell and Plutus. Haskell is used to code Cardano while Plutus is being used to write smart contracts. Both belong to the family of functional programming languages. Functional programming languages are easier to verify mathematically. This makes them a more secure approach for writing smart

contracts. This specific platform has a control layer that separates use of smart contracts from the currency it uses. [75]

Rootstock RSK blockchain is developed to be compatible with Ethereum but bitcoin cryptocurrency. Main idea behind its invention is to give bitcoin blockchain smart contract capabilities. Rootstock interacts with bitcoin using sidechain technology. It is connected to bitcoin via two-way peg. This means users can mine currency in bitcoin and get an equivalent amount in Rootstock. This cryptocurrency can be used to deploy or interact with smart contracts and dApps on rootstock blockchain. At its core, rootstock uses turing complete resource accounted virtual machine which is compatible with Ethereum blockchain. Bitcoin provides RSK with proven security, wide distribution and awareness. RSK uses POW algorithm as consensus mechanism. RSK uses a technique called merge mining to use the same hashing rate as bitcoin blockchain. Basically RSK helps users to enjoy the benefits of bitcoin blockchain with extra functionalities of smart contracts. [76]

Below table is the summary of differences between major competitors of smart contract platforms.

	Ethereum	Eos	Cardano	RSK
Native Token	ETH	EOS	ADA	RBTC
Company	Ethereum Foundation	Block.one	IOHK and Emurgo	RSK Labs
Base platform	Ethereum	Eos	Cardano	Bitcoin
Consensus	POW now Casper later	DPoS	Ouroboros	Merge-mining
Transaction per second	15-25 tps	Max: 4000 Avg: 50	50-250 (during testing)	15-25 tps
Block time	10-20 seconds	0.5 seconds	<20 seconds	15-30 seconds
Smart Contract Language	Solidity	WASM	Plutus	Solidity

[77]

Table 7 Difference between smart contract platforms

Blockchains like Lisk, Neo and Stratis are also being developed for deploying smart contracts. Neo is an open source blockchain backed by Chinese government. That's why it

is also referred to as china's blockchain. Neo supports the use of imperative programming languages like python, C++ and Java. Neo uses dBFT as consensus algorithm. Lisk uses Javascript as programming language for smart contracts and applications developed in Lisk are decentralized. Lisk offers a wide range of services like hosting, storage nodes and computing nodes which are very similar to cloud computing services. [78]

2. Ethereum Platform

2.1 Proof of Stake: -

Implementing Proof of work uses a lot of energy. Bitcoin implementing POW uses a lot of electricity. It has been estimated that electricity used in implementing POW can almost be used to power up a whole country like Hungary or New Zealand. Mining new coins use a lot of power. Moreover, this strategy makes Blockchain a kind of centralized technology. Because a lot of people around the world are using mining pools to mine new coins. Adding a new block to a chain gives a certain reward in terms of coins. So people are using a lot of systems that have huge computing power all together so that they can earn rewards. This is making Blockchain centralized.

To avoid this, researchers have come up with new algorithm called proof of stake. POS have validators as compared to miner in POW. The POS algorithm uses less energy because it doesn't require a lot of cryptographic calculations to find POW to validate each block. We don't need expensive mining equipment to let a node validate as in POW.

In POS, algorithm uses a pseudorandom method instead of random to validate a new block. Basically in POS, any node who wishes to validate a new block must have a certain supply of coins. Concerned node has to deposit those coins as a stake in network. This stake can be taken as security deposit. Probability of node which is to be selected to validate a new block depends upon the amount of coins deposited. If a node owns A amount of coins and there are total B coins in network, the probability is A/B . If a node stops being a validator it will get its transaction fees and the stake after sometime. POS algorithm also takes into account the possession time of coins in order to prevent rich nodes to be selected every time. As a reward, the validator will get a certain amount of interest on the basis of amount deposited after adding new block. [18]

Implementing POS algorithm also prevents the attack of 51%. Because as compared to POW, holding 51% stake in POS is quite expensive and nearly impossible. And if the attacker owns 51% of coins in chain, that means if something happens to chain he will lose his own coins as well. In POS, validators do minting and forging not mining like POW.

However, there are some disadvantages to POS algorithm too. It may seem that in POS algorithm rich nodes will get even more richer because they have more coins and they will invest more, earn more and it will lead to centralization. But in reality it's more different and better than POW. In POW, rich people can enjoy economics at scale. Price they pay for mining and electricity doesn't go in a linear fashion. For example, if 1Kwh=\$1 it doesn't necessarily mean the 1000Kwh will be equal to \$1000. It can be equal to \$800. So more they invest, better prices they can get. And in POS validators will lose a part of stake if they choose a fraudulent block.

POS algorithms suffer from a problem of nothing at stake. A participant who has nothing at stake has no reason not to behave badly. In presence of multiple chains, nodes will allocate their stake uniformly because no one encourages to put their stake in legitimate chain. Therefore, nodes vote on multiple chains, supports multiple forks so that they can maximize the chances of winning the reward. Because unlike in POW, miners in POS doesn't have to invest in electricity and mining power. This can cause to composing of multiple blocks in parallel chains. Correct implementation of POS would be required to avoid this problem.

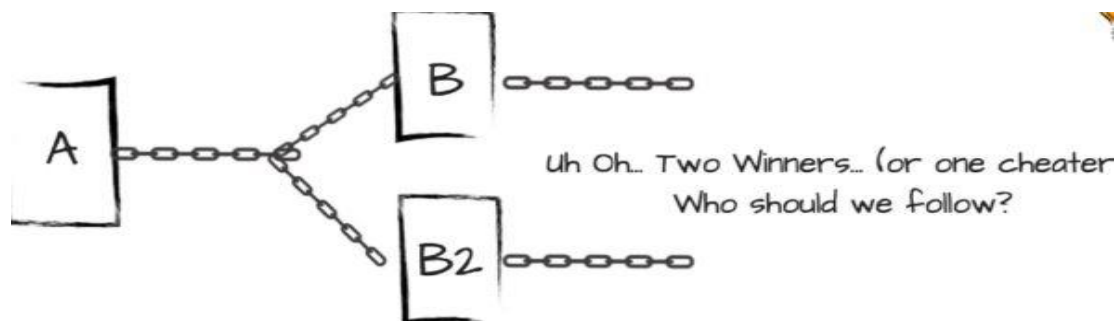


Figure 2.1 Depiction of how POS works -1

[19]

In the above case, two new blocks have been proposed. Now participants have to choose one of the two chains. But the participant can choose to follow both chains.

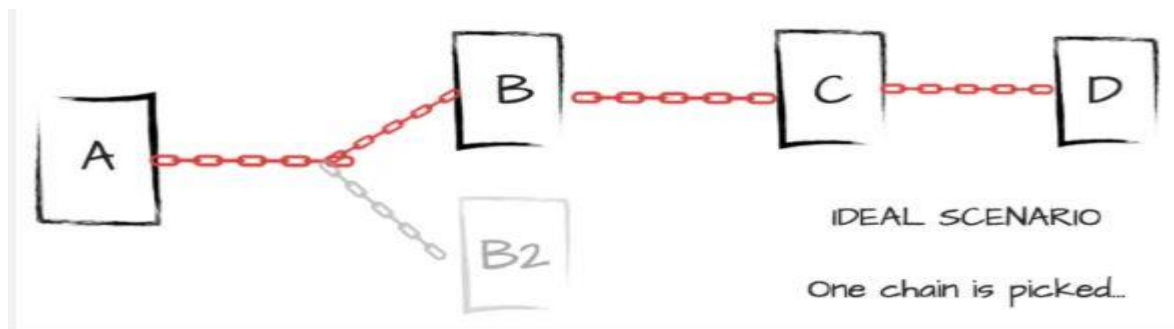


Figure 2.2 Depiction of how POS works - 2

Because he has deposited stake. Since the chain has been forked, his deposit exists on both new chains. So the validator will pick both chains in order to win reward on both chains.

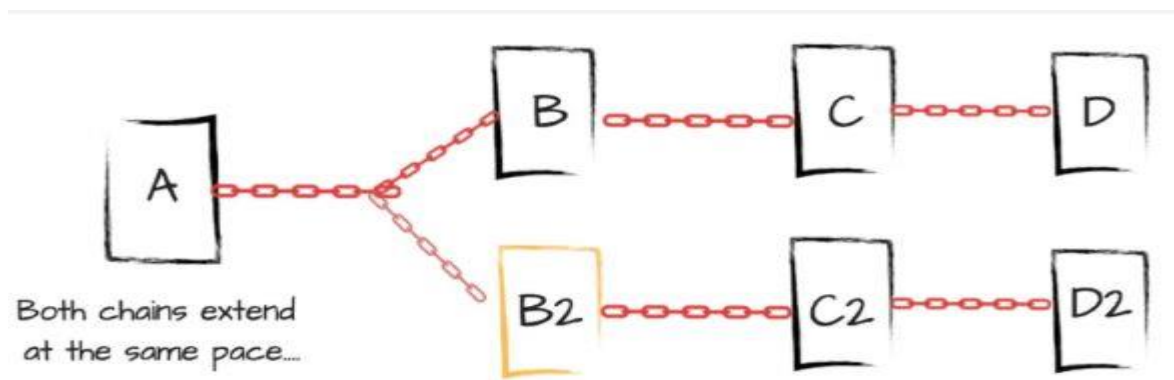


Figure 2.3 Depiction of how POS works - 3

So the resultant chain will look something like this. Now everyone is voting for both sides so both chains extend at same pace. This compromises the security of Blockchain. Any malicious actor can fork the chain and get away with double spending. He will keep on doing certain things until the chain with bad transaction outpaces the first one. The bad transaction will then be officially accepted. [19]

2.2 Relation between Ether and Gas: -

Just like Bitcoin being the cryptocurrency for Bitcoin Blockchain, Ether is the cryptocurrency for Ethereum. All the trades and goods sold on Ethereum is being done in Ether. It is used to purchase gas which is used as crypto-fuel. Ether is also rewarded to miners for doing intense computational practices for successfully adding new blocks to chain. Ether can be easily converted to dollars or other valuable currencies at crypto exchanges.

Ethereum has a metric system of denominations used units of Ether. The smallest denomination of Ether is Wei and all else is the multiples of Wei. Following is the list denominations and their values in Wei.

Units: Weis

'wei': '1',
'kwei': '1000',
'ada': '1000',
'femtoether': '1000',
'mwei': '1000000',
'babbage': '1000000',
'picoether': '1000000',
'gwei': '1000000000',
'shannon': '1000000000',
'nanoether': '1000000000',
'nano': '1000000000',
'szabo': '1000000000000',
'microether': '1000000000000',
'micro': '1000000000000',
'finney': '1000000000000000',
'milliether': '1000000000000000',
'milli': '1000000000000000',
'ether': '1000000000000000000',
'kether': '1000000000000000000000',
'grand': '1000000000000000000000',
'einstein': '1000000000000000000000',
'mether': '1000000000000000000000000',

blocks to chain, they are rewarded in ether coins. Blocks are added in Ethereum approximately every 15 seconds, so the amount of ether increases correspondingly.

Transferring Ether: After the successful generation, ether is transferred to miner's external account. He can transfer it further to contract account using Ethereum wallet or using programmes.

Storing Ether: You can manage ether in accounts by different methods, each presenting a trade-off between convenience and security. Easy way is to save it in Ethereum wallet and transfer it conveniently when required. Another way is to write the private key on hard paper and storing it. Hardware wallets are more secure than online because they are literally offline.

Wallet type	Convenience	Security
Desktop wallet	High	High
Mobile wallet	High	Low
Online wallet	High	Medium
Hardware wallet	Medium	Very high
Paper storage	Low	Very high

Table 3 Different Wallet Types

[37]

Exchanging Ether: Ether can be sold for a conventional currency like US dollars, euro, pounds and yen. It doesn't get transferred between accounts for free, mostly it is transferred in return of goods and services mentioned in smart contracts. You can buy ether in exchange for conventional currencies as well but it's more secure and effective to handle such transactions online. [35]

Below is the diagram describing ether life cycle:

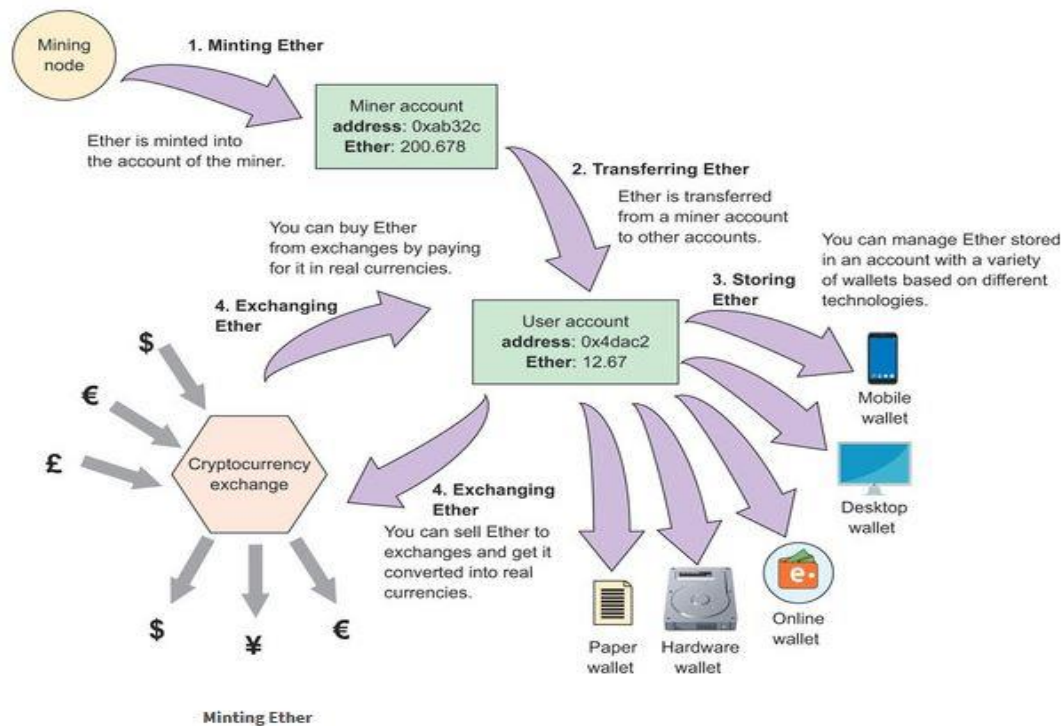


Figure 2.4 Ether Life Cycle

[36]

GAS: -

Gas is the internal currency of Ethereum. The concept of Gas is almost the same as the electricity we consume at our houses. One actual difference is that the originator of the transaction sets the price of gas. The amount of gas is related to the amount of computational resources that Ethereum network spends while performing the change or transaction. Ether is usually paid as fees for any changes in decentralized network to cover up processing fees. But the price of ether doesn't remain constant. It fluctuates every day, could be high on some days and really low on others. Because of that, people willing to make changes in the network, will wait for prices of ether to fall down. That is not an ideal situation for Ethereum network. The resource utilization in Ethereum network is already made permanent in terms of gas. This is called gas cost. There is also a term called gas price which is kept higher when ether price is lower and kept lower when ether price is higher. [38]

Below are some low level instructions on Ethereum cost in terms of gas chart:

Operation	EVM op code	Gas cost
Addition, subtraction	ADD, SUB	3
Multiplication, division	MUL, DIV	5
Comparison	LT, GT, SLT, SGT	3
Load word from memory	MLOAD	3
Store word to memory	MSTORE	3
Load word from storage	SLOAD	200
Storing word into storage	SSTORE	>5000
Contract creation	CREATE	32000

Table 4 Instructions and their gas costs

[39]

The main reason why Ethereum have a concept of gas comes into being with smart contracts. So if a hacker tries to run a malicious code that contains instructions to create a loop, and run that smart contract on Ethereum virtual machine. That code will run forever. This is how hacker can start a denial of service (DoS) attacks. In order to fix this loop problem, gas is used. Every transaction on Ethereum requires a certain amount of gas. Every account in Ethereum has ether. So for example, if you are going to store a word in memory, MSTORE, that is the gas of 3. If you are going to load a word from storage, that is gas of 200. Storing a word in permanent storage is expensive, that is nearly around 5000. All these operations are in byte code form. So when you compile a contract, different type of operations run of virtual machine. That all combined give you the gas to run that smart contract. So the formula used for calculating Transaction fees in ether = number of gas units consumed * price per unit of gas.

As mentioned above there is a concept of GasLimit and GasPrice. GasLimit is the gas purchased by the sender. It is valued in Wei. Wei is a small fraction of ether. Normally 4.7 million Wei is used for running a contract. There is a limitation for GasLimit as well. This means you cannot run that contract forever; it would stop when it runs out of gas. GasPrice is set by the transactor. It's a criteria miner looks into for running a program. You have to set the GasPrice in contract you are going to run. GasPrice is normally valued in GigaWei's, but it depends on the situation. If a GasPrice is kept low and miner doesn't have many programs to run, then he will run your program. And If he already has a lot of programs to run, then GasPrice will go higher, and your program will not be able

to run. This is all a supply and demand situation. If you are in a hurry, you can set the GasPrice higher so that miner will include your transaction. If you are not in a hurry, you can just set it at a nominal range, so some miners will eventually include your transaction. The more GasPrice you set, the more Wei you use. Wei equates to ether, so in turn you are actually using few cents to run a program.

There are few specific properties of the Ethereum network like if Gas runs out while the contract is running, all the processing will be reverted back. But transactor will still be charged. You would not be able to store anything in Blockchain. So it's best to estimate the appropriate value of gas before running the contract. If the transaction is completed and there is still some gas left, it would be returned to sender's account. Denial of service attacks are prevented in this way. That's how the concept of gas is used to make Blockchain more immutable and secure. [40]

2.3 EVM (Ethereum Virtual Machine): -

Ethereum virtual machine provides an execution runtime for Ethereum network. EVM can be called as the operating system of Ethereum network. It enables users to run codes written in smart contracts on Ethereum Blockchain. In fact, the main job of EVM is to execute the code line by line on Blockchain. It can be used to access accounts on Blockchain, view transactions happening in real time. It doesn't have access to the whole ledger but it can provide information about recent transactions.

EVM is Turing complete. Any programming language computationally equivalent to Turing machine can be referred to as Turing complete. Turing machine is invented by Alan Turing, which is used to stimulate the logic in today's computers. Turing machine should be able to solve any mathematical computation problem. Any programming language is called Turing complete if it satisfies two conditions overall: first it has a way to read and write storage variables. Second it has a form of a conditional jump like while, If, goto. For example, if you write a program that takes 20 numbers and performs addition on them. If a programming language can run it, it is turing complete or otherwise it's turing incomplete. Languages like C, C++, Java are all turing complete. Languages like SQL, HTML are not turing incomplete. Unlike other virtual machines, EVM is bounded by value of gas rather than memory or time. In EVM, memory grows as needed. [48]

Ethereum virtual machine reads bytecode from smart contract compiled by high level compiler like remix. Instructions for EVM on how to execute the bytecode comes from the sequence of opcodes, data and static data values that can be used during execution. All Ethereum opcodes are one-byte long. There are a total of 129 opcodes in EVM. Value of opcodes are represented in Hexadecimal which ranges from 0x00 to 0xff. Ethereum virtual machine reads the bytecode by incrementing a program counter. Opcodes like PUSH, JUMP and JUMP require program counter to be incremented more than 1 and these opcodes have certain constraints as well. [47]

Table below shows the most prominent opcodes in EVM: -

STOP	0x00	Halts execution
ADD	0x01	Adds two values
MUL	0x02	Multiplies two values
SUB	0x03	Subtracts two values
DIV	0x04	Divides two values
EXP	0x0a	Exponential Operation
ADDRESS	0x30	Used to get the address of currently executing account
BALANCE	0x31	Used to get the balance of given account
ORIGIN	0x32	Used to get the address of sender of original transaction
CALLER	0x33	Used to get the address of initiator of execution
CODESIZE	0x38	Retrieves the size of running code in current environment.

GASPRICE	0x3a	Retrieves the gas price specified by initiating transaction.
BLOCKHASH	0x40	Gets the hash of one of the 256 most recently completed blocks.
COINBASE	0x41	Retrieves the address of beneficiary set in the block.
GASLIMIT	0x45	Gets the gas limit value of block.
POP	0x50	Removes item from stack.
MLOAD	0x51	Used to load a word from memory.
MSTORE	0x52	Used to stores a word to the memory.
CREATE	0xf0	Used to create a new account with associated code.
CALL	0xf1	Used to initiate a message call into an account.
RETURN	0xf3	Stops the execution and returns the output data.
SUICIDE	0xff	Halts the execution.
GAS	0x5a	Retrieve the available gas amount.
SLOAD	0x55	Saves a word to the storage.
CODECOPY	0x39	Copies the running code from current environment to memory.
PUSH 1.....PUSH 32	0x60...0x7f	Used to place N right-aligned big endian byte items on the stack.

Table 4 Opcodes in EVM

EVM is based on a stack-based architecture. Every computation in EVM is done in bytecode. Word size of EVM is 256 bits. EVM will store all the results of executions as per bytecode. Size of every item in stack is 256 bits, if anything is smaller than 256 bits it will be padded with extra zeros. Stack size of EVM is 1024 elements and it is based on LIFO (Last in First Out). Memory in EVM is volatile while storage is permanent. Arguments are called as a result of running code. Below diagram shows EVM block structure.

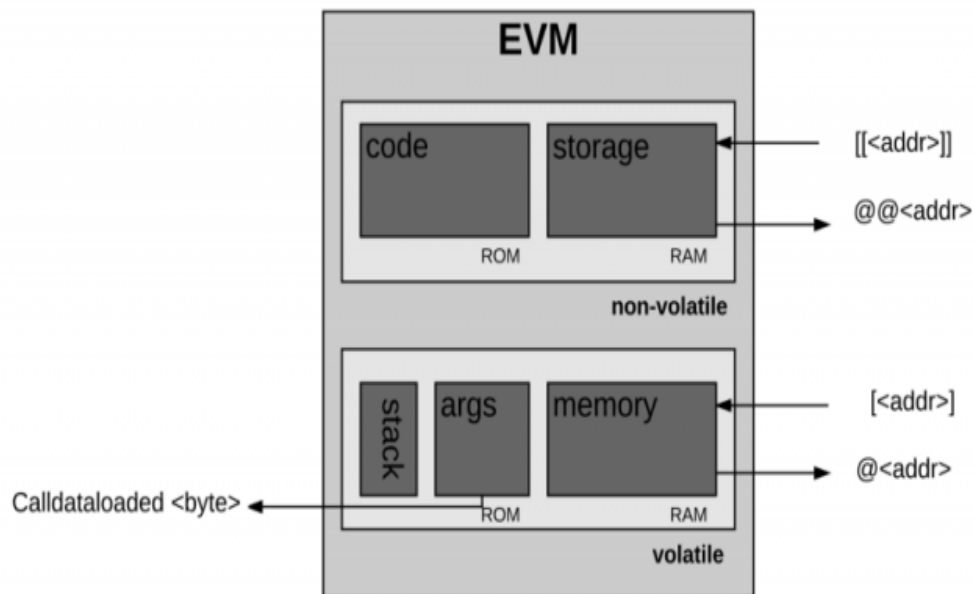


Figure 2.5 EVM Block Structure

[49]

As discussed above EVM has two types of storage available. One is memory and other is storage. Storage is permanent and it's a key value store. While memory is in form of byte array which finishes when codes is done. Any program code written in Ethereum virtual machine is stored in virtual read only memory (virtual ROM). EVM is an isolated sandbox environment. Code running in there won't get any resources from outside.

The diagram below shows the V-ROM where code is stored. Then by using CODECOPY Opcode it is moved into main memory. Main memory then executes the instructions one by one by incrementing a program counter. The program counter and EVM stack are

updated accordingly with every instruction. Opcodes PUSH and POP are used for LIFO in EVM stack. EVM stack is 32 bytes in size. [50]

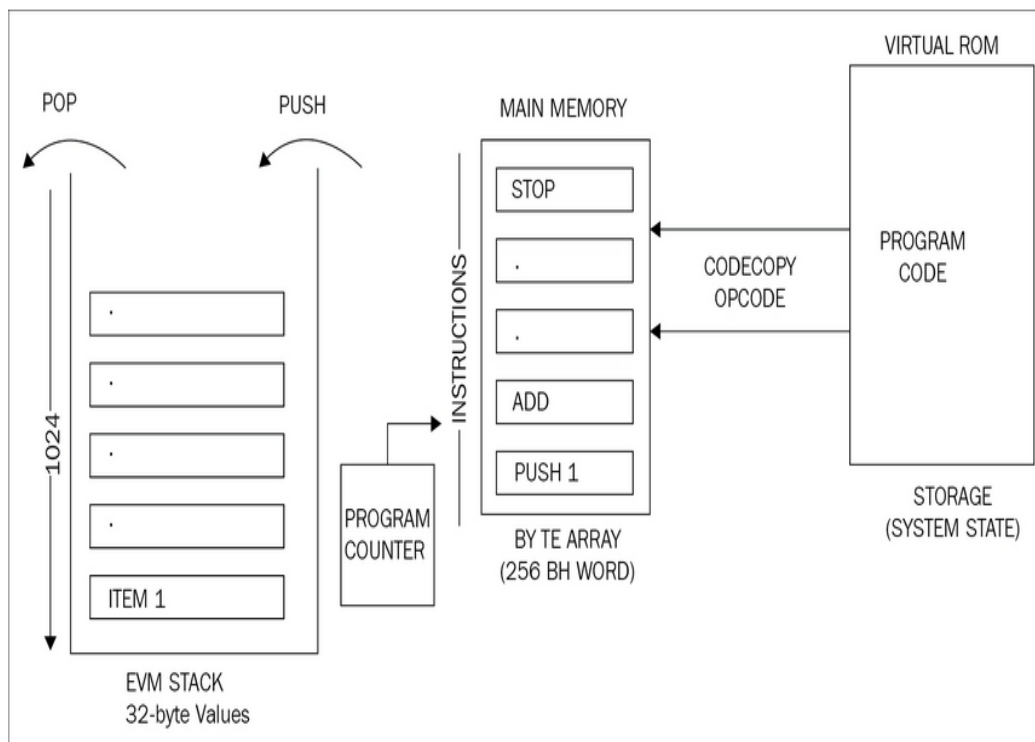


Figure 2.6 Block Diagram of V-ROM and Stack of EVM

[51]

Some key basic elements are required in order to execute the code in Ethereum network. By a rough assessment, they are around nine shown in diagram below. Firstly, you need address of the account that owns the code along with address of sender of transaction. Gas price should be listed in a transaction. Next requirement is input data. Input data is in form of byte array or transaction. In case of message call, it is byte array otherwise it's transaction data. Next thing is address of initiator of transaction. Then the value in Wei. This is the transaction value if the execution agent is transaction. Executed code should be presented as byte array for the iterator function to pick up in each cycle. Other thing required is block header. It is the number of opcodes CALLs and CREATEs in transaction. In addition to this, system state and remaining gas are also provided to the environment. [52]

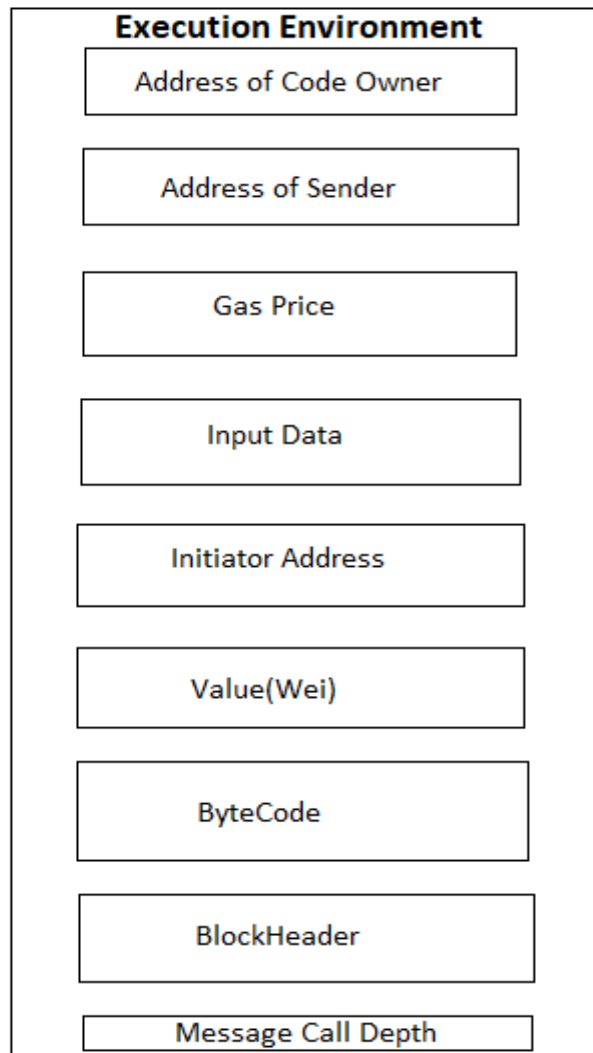


Figure 2.7 Execution Environment in Ethereum Blockchain

2.4 Ethereum Accounts: -

Any transaction in Ethereum requires an account. Accounts are a fundamental part of Ethereum Blockchain. Accounts have a particular state and this state is updated as a result of any transaction in Blockchain. There are two types of accounts in Ethereum.

- Externally owned accounts
- Contract Accounts

Externally owned accounts have ether balance. It is able to send transactions and has no code associated with it. EOA's are controlled by private keys. Transactions in EOA can mean to transfer ether or trigger contract code. Contract accounts are controlled by code

and have private keys associated with it. Contracts accounts also have ether balance. These accounts have the ability to trigger and execute code in response to transaction or message. Contract accounts when executed performs operations of arbitrary complexity and manipulate its own persistent storage, call other accounts.

Contracts in Ethereum act like autonomous agents in Ethereum environment. Contracts are not waiting to be compiled or fulfilled like other environments. They will always execute a specific code when called or poked and will have their control on ether balance, value store to keep track of variables.

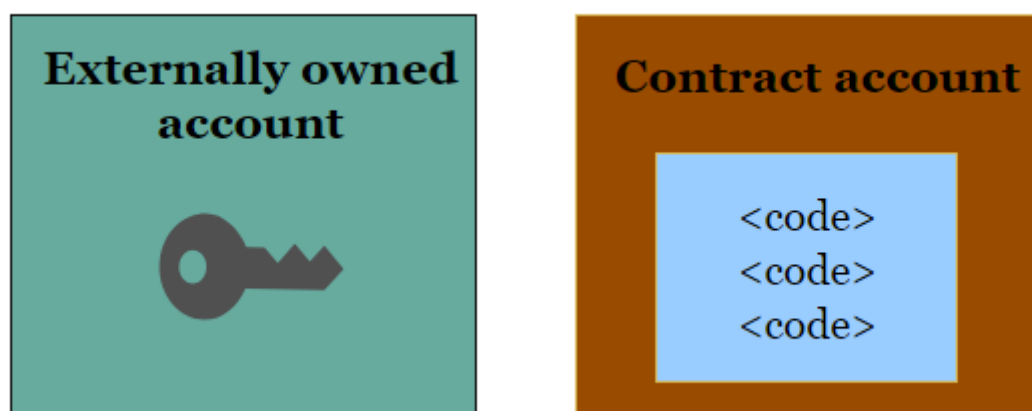


Figure 2.8 Types of accounts

There are some advantages of contract accounts over externally owned accounts like contract accounts can list incoming transactions. Contract accounts can work as multisig accounts, multisig accounts can have specified daily limits by user and if the daily limit exceeded it will require multiple signatures. However, EOA are free to create and operate while contract accounts will have a certain cost. Another disadvantage of contract accounts is that they can't initiate new transactions on their own. EOA's can send message to other EOA's or contract accounts by using private key. Message from EOA to EOA is just a value transfer but a message to contract account invokes a contract which requires to spend some tokens or write new contract. Contract account can only fire transactions they receive from other contract account or EOA. [53]

Regardless of type or account, state of account will consist of four different components:

Nonce: - It is a counter that is used to make sure that every transaction is only processed once. Nonce represents a scalar value sent from the particular address. For EOA, nonce is number of transactions sent from account's address while for contract account, nonce is number of contracts created by account.

Balance: - Balance is the scalar value used to represent number of Wei owned by account. There are $1e+18$ Wei per ether.

StorageRoot: - This is the 256-bit hash that encodes the storage contents of the account.

codeHash: - codeHash contains the hash of Ethereum virtual machine for the code that is being executed. Message call received for this account is immutable and can't be changed once constructed. All such code fragments are contained in same database under their relatable hashes for later retrieval. For EOA, this is the hash of empty string while for contract account this is the code that gets hashed and get stored as codeHash. [54]

3. Smart Contracts

3.1 Solidity: -

Solidity is the programming language invented specifically for writing smart contracts. Solidity has close resemblance with C and JavaScript. It's an object-oriented programming language like Java. Solidity is a strongly typed language while JavaScript is dynamically typed. It's a case-sensitive language. This means `dog` is different than `DOG` or any other versions of `dog`. Solidity files have an extension `.sol` and they are in human-readable context. Solidity code can be opened in applications like Notepad.

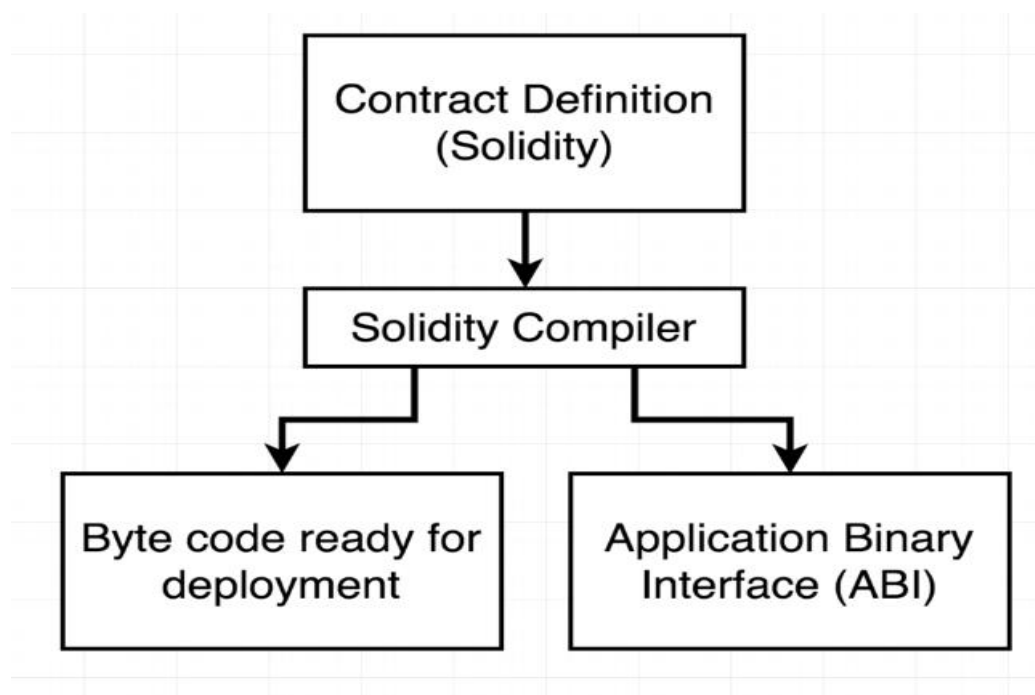


Figure 3.1 Block Representation of how compiler works

[55]

When a user writes a contract in Solidity, it is mentioned as Contract definition. Solidity as a programming language is very easy to read, understand and interact with. In background, actual contract doesn't interact with Blockchain. Contract definition when fed into specialized Solidity compiler will generate two separate files. First file is the byte code that is actually deployed in Ethereum network. Second file generated is ABI Application binary interface. ABI is a key component for applications to interact with deployed contract on Ethereum network. ABI works as an interface between JavaScript

console and byte code. ABI helps JavaScript to understand bytecode. ABI is actually easy to read for humans and it will give user an idea of how to interact with the contract.

Solidity file normally consists of four main components: -

- Pragma
- Comments
- Import
- Contracts/ library/ interface

Pragma: - Pragma is the first line of solidity code that specifies the compiler version being used for this code. As solidity is continuously changing language, so whenever a new feature is introduced it is integrated with newer version. Pragma is not mandatory while writing a code in solidity but it's a good practice to do so. Normal solidity version looks like 0.4.19. It consists of a major build and minor build number. 4 is the major build while 19 is minor build. There are few or little changes in minor build while there can be significant changes in major build versions.

Comments: - Solidity like other programming languages provide the facility of comments. There are three different types of comments in solidity. 1) Single-line comments 2) Multi-line comments 3) Ethereum Natural Specification NatSpec

NatSpec is used for documentation purposes. It is recommended that solidity contracts are fully annotated using NatSpec. NatSpec includes the formatting for comments that smart contract author will specifically use and a solidity compiler can understand. [55]

Single line comments are represented by a double forward slash //, multiline comments are represented by /* and */. NatSpec are is donated by /// or either by /** and */.

Import: - Import helps us write a modular solidity code. By using Import, we can import other solidity files and we can access those codes within this solidity code. Syntax of import is `import <<filename>> ;` Referring to a file in solidity is very similar to Linux bash way. Forward slash is used to separate directories and . is used to show the present location of file. [56]

Contracts: - Contract in solidity is similar to class in object-oriented languages. Contracts, interfaces and libraries are defined at global level. Keywords contract,

interface and library are case-sensitive in nature. Contract code consists of variables and functions which can read and modify like in traditional programming language. [58]

Below figure is a simple example of solidity code in which users can deposit some value and check their balance.

Example 1:

```
pragma solidity ^0.4.17; 1
contract SimpleDeposit { 2
    mapping (address => uint) balances; 3
    4
    event LogDepositMade(address from, uint amount); 5
    6
    modifier minAmount(uint amount) { 7
        require(msg.value >= amount); 8
    } 9
    10
    function SimpleDeposit() public payable { 11
        balances[msg.sender] = msg.value; 12
    } 13
    14
    function deposit() public payable minAmount(1 ether) 15
    { 16
        balances[msg.sender] += msg.value; 17
        LogDepositMade(msg.sender, msg.value); 18
    } 19
    20
    function getBalance() public view returns (uint 21
        balance) { 22
        return balances[msg.sender]; 23
    } 24
    25
    function withdraw(uint amount) public { 26
        if (balances[msg.sender] >= amount) { 27
            balances[msg.sender] -= amount; 28
            msg.sender.transfer(amount); 29
        } 30
    } 31
}
```

Figure 3.2 Sample Code - 1

[57]

In example 1, line 1 shows compiler version. Line 2 defines a contract SimpleDeposit. State variable is defined in line 3 followed by event definition in line 5. Line 7 is definition of modifier while line 12 is constructor. Line 16 shows the actual contract function. Constructor function SimpleDeposit is run during the contract creation and can't be called afterwards. State of this contract is stored in mapping called balances. All the

remaining functions serve as an interaction and may be called by users and contracts alike. Line 15 defined withdraw function. This function works to subtract the amount from sender account and line 21 getBalance () function returns the remaining value to sender account by checking the balances mapping. [61] Deposit functions line 16 uses the balances mapping by adding the amount sent along the transaction to the sender's balance, while keeping a check on a modifier that only ether is sent. [62]

Solidity as a programming language is primarily designed to write smart contracts in Ethereum. Smart contracts are mainly composed of constructs, variables and functions. An ideal contract consists of following multiple constructs: -

- State Variables
- Structure definitions
- Modifier definitions
- Event Declarations
- Enumeration definitions
- Function definitions [60]

In the following example 2, every construct mentioned above has been used. In this particular contract, each construct in turn consists of multiple constructs.

Solidity is classified into two data types based on the way they are assigned to variables and stored in EVM. The types are values types and reference types. Value type maintains an independent copy of variable and changing the value in one variable doesn't affect other variable. While reference type keeps the updated value of variable.

Example 2:

```

pragma solidity 0.4.19;

//contract definition
contract generalStructure {
    //state variables
    int public stateIntVariable; // vriable of integer type
    string stateStringVariable; //variable of string type
    address personIdentifier; // variable of address type
    myStruct human; // variable of structure type
    bool constant hasIncome = true; //variable of constant nature

    //structure definition
    struct myStruct {
        string name; //variable fo type string
        uint myAge; // variable of unsigned integer type
        bool isMarried; // variable of boolean type
        uint[] bankAccountsNumbers; // variable - dynamic array of unsigned integer
    }

    //modifier declaration
    modifier onlyBy(){
        if (msg.sender == personIdentifier) {
            _;
        }
    }

    // event declaration
    event ageRead(address, int );

    //enumeration declaration
    enum gender {male, female}

    //function definition
    function getAge (address _personIdentifier) onlyBy() payable external returns (uint) {

        human = myStruct("Ritesh",10,true,new uint[](3)); //using struct myStruct

        gender _gender = gender.male; //using enum

        ageRead(personIdentifier, stateIntVariable);
    }
}

```

Figure 3.3 Sample Code - 2

[59]

State Variables: -

Solidity has two types of variables, state and memory variables. Variables in normal day programming direct to a location in storage containing values. These values can be changed during the running of code.

State variables in solidity are indefinitely stored in Blockchain. Memory allocated to state variables is statistically assigned and it cannot be changed afterward. These variables are

not called in any function of the contract. Each state variable has a data type with it. These data types help with the memory allocation of variables. Data types in solidity are following: -

- Bool
- Uint/int
- Bytes
- Address
- Mapping
- Enum
- Struct
- Bytes/string

Qualifiers are also defined with state variables. Qualifiers define the function these variables can perform. Major qualifiers are: -

- Internal
- Private
- Public
- Constant

Structure: -

Structure or struct consists of multiple variables with different data types in them. But structures don't contain any code. Basically structure helps make user favorable data types. For example, if a grocery clerk wants to store different products from one brand. He will define the structure of that brand and will define those variables as products from that brand.

Syntax for defining structure is as follows: -

```
//structure definition
struct myStruct {
    string name; //variable fo type string
    uint myAge; // variable of unsigned integer type
    bool isMarried; // variable of boolean type
    uint[] bankAccountsNumbers; // variable - dynamic array of unsigned integer
}
```

[63]

Events: -

Events in solidity are used for informing the calling application about current state of contract being deployed on EVM. Events keep a track on changing state of contracts and let the calling applications know about the change. So instead of applications keeping a check on changing state of contract, contract informs them by means of events. [64]

Events are announced on global level. They are defined by keyword event followed by an identifier and parameter list. An example is displayed below.

```
// event declaration
event ageRead(address, int );
```

Enumeration: -

Defining an enum is not mandatory in solidity. These are user defined and contain human readable names for a set of constants called member. Numbers or Boolean variables are not permitted as members in the enumeration. Enums are useful when there is a need to specify the current state of contract. [65] Below are the examples of few declarations of enum in solidity.

```
enum Direction { Left, Right, Straight, Back }
enum ContractType { permanent, temporary, apprentice }
enum State { Created, Locked, Inactive }
enum Status { ON, OFF }
```

[66]

Enum declaration doesn't have to be ended with semi-colon.

Functions: -

Function is declared by keyword function. Parameters provided are optional. Data types are in those parameters. Functions have the ability to read and write the value to or from the state variables. Transaction is created when a functions is invoked in a smart contract.

Functions have qualifiers associated with them like variables. There are two types of qualifiers, visibility and additional.

Visibility qualifiers are as follows: -

- Public
- Internal
- Private
- External

Additional qualifiers are: -

- Constant
- View
- Pure
- payable

3.2 How are Contracts Deployed: -

So the main question here is how can people connect with Ethereum network? How can they send money or transfer data on Ethereum? Well, there are two groups of technologies that helps us connect with Ethereum. First technology is for the developers called Web3.js. It's the main java script library for communicating with Ethereum Blockchain. Web3.js is sort of like a portal that lets us send money or store data or whatever we want to do on network. Actual applications are built on web3.js that talk to the network through code.

The second technology is for the consumers. People who don't have any programming background. This technology offers two solutions, MetaMask and Mist. MetaMask is the browser extension that lets people interact with Ethereum. MetaMask acts like a web client. If you are accessing any website using decentralized storage system, you will have to use MetaMask to connect to Ethereum in order to use tokens. As MetaMask runs in the browser, it doesn't download the entire chain data as in the case of geth node, instead it stores it centrally and help users connect to their store using browser. Its' by far the most common way of interacting with Ethereum. [43]

Mist is the full featured web browser that is intended to browse Ethereum applications. You can develop and execute different decentralized applications and projects. It is still in developing stages, so not a lot of people prefer to use this browser. There is a Mist wallet as well that is used to store, send receiver cryptocurrency. It runs of local user's computer, so you have to download and install it on your machine.

MetaMask Setup: -

We will be dealing with MetaMask only. To install MetaMask in your browser, access the following site. <https://metamask.io/> . Install the extension desired on specific browser like chrome or Firefox. I am using Firefox so I will be installing Firefox extension. It would be something like displayed below here.

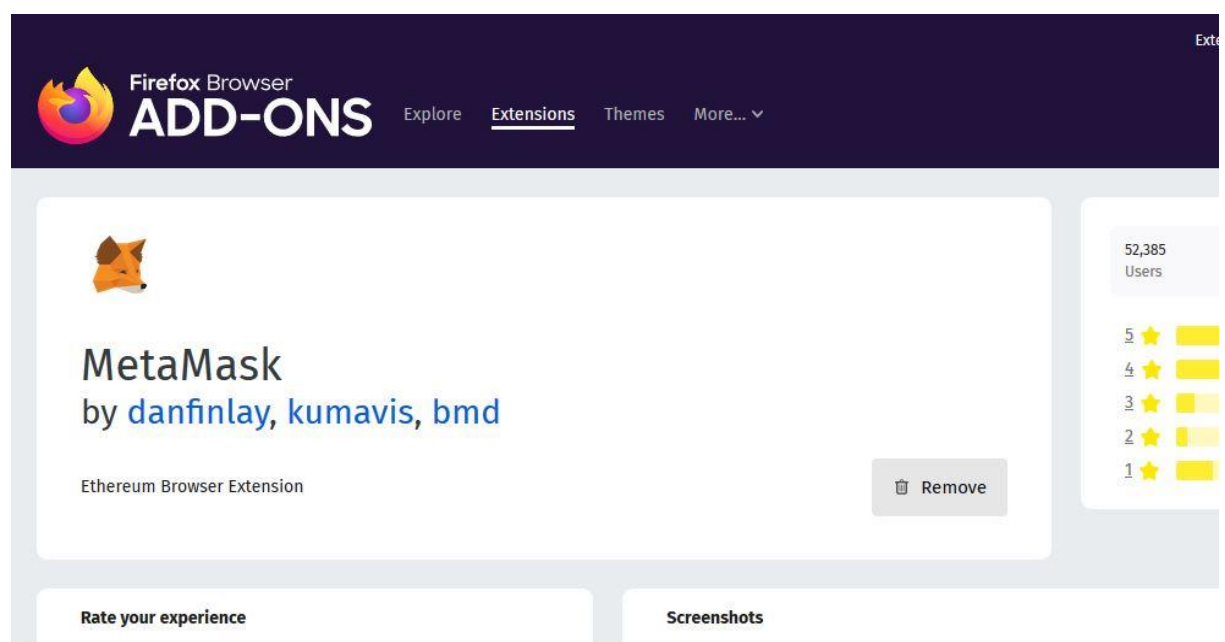


Figure 3.4 MetaMask Setup in FireFox - 1

After installing extension, you will an icon of MetaMask at upper right side of browser. When you click that icon, you will get an option to whether create a new account on Ethereum. Or if you already have one just import that one here.

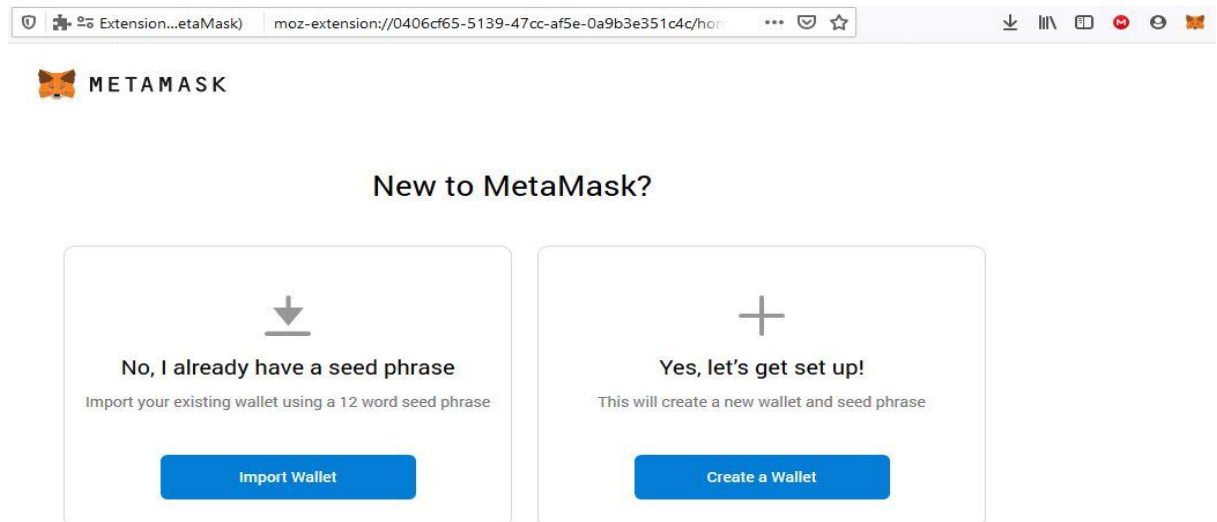


Figure 3.5 MetaMask Setup in FireFox - 2

Since it's my first time using Ethereum, I will be setting up a new wallet. After clicking create a new wallet, a message will display about security policy of MetaMask. We will

agree with that.



Help Us Improve MetaMask

MetaMask would like to gather usage data to better understand how our users interact with the extension. This data will be used to continually improve the usability and user experience of our product and the Ethereum ecosystem.

MetaMask will..

- ✓ Always allow you to opt-out via Settings
- ✓ Send anonymized click & pageview events
- ✓ Maintain a public aggregate dashboard to educate the community
- ✗ **Never** collect keys, addresses, transactions, balances, hashes, or any personal information
- ✗ **Never** collect your full IP address
- ✗ **Never** sell data for profit. Ever!

No Thanks

I agree

This data is aggregated and is therefore anonymous for the purposes of General Data Protection Regulation (EU) 2016/679. For more information in relation to our privacy practices, please see our [Privacy Policy here](#).

Figure 3.6 MetaMask Setup in FireFox - 3

After that we will be asked to create a password. After creating a password, you will be displayed with mnemonic. Save this mnemonic on a text file separately. Then you will get to the screen of your Ethereum account.



< Back

Create Password

New Password (min 8 chars)

Confirm Password

I have read and agree to the [Terms of Use](#)

Figure 3.7 Password Creation in MetaMask



Secret Backup Phrase

Your secret backup phrase makes it easy to back up and restore your account.

WARNING: Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.

kit vacuum drink plug immune fee
cause barrel chat cotton catch
sand

Remind me later

Next

Tips:

Store this phrase in a password manager like 1Password.

Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.

Memorize this phrase.

Download this Secret Backup Phrase and keep it stored safely on an external encrypted hard drive or storage medium.

Figure 3.8 Backup Procedure in MetaMask

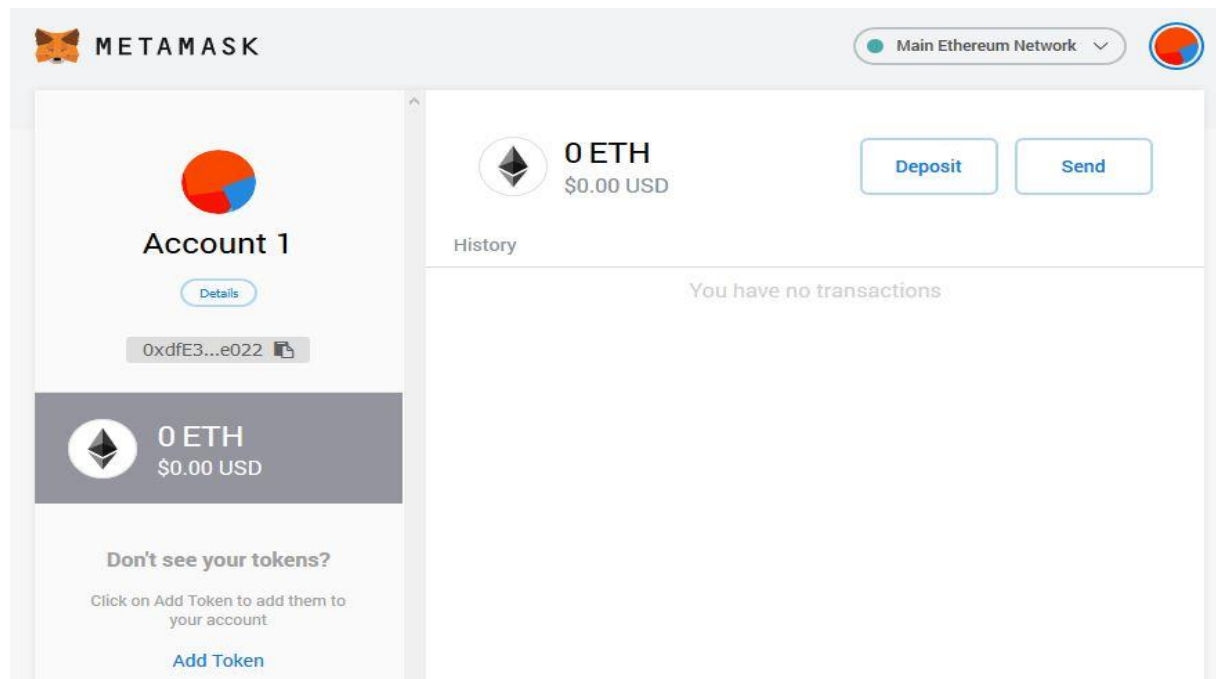


Figure 3.9 Main Window of MetaMask

On the upper right side of page, you will see a scroll down bar. By default, main network is selected. Real Ether that means real money is used on main network. I won't be using main network. Further down, you will see many test network options. We will be practicing with rinkeby network. Localhost 8545 corresponds to your local machine. This means that if you are running a geth node on your machine, you can use that option. Custom RPC is used to connect to remote network.

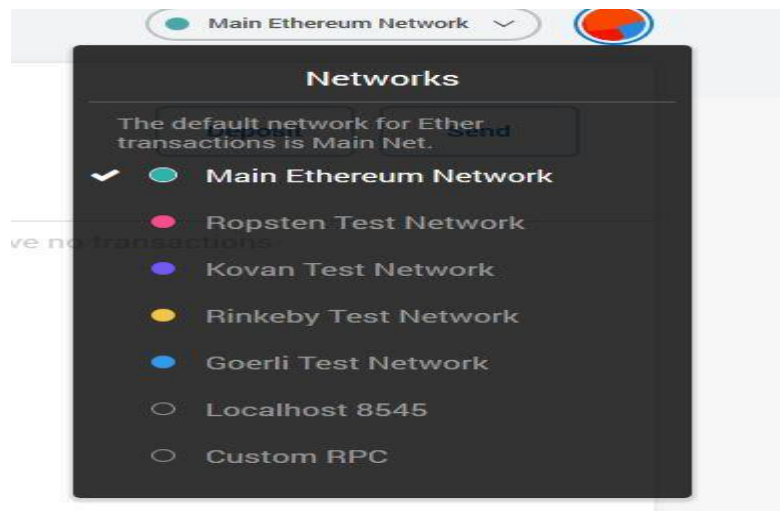


Figure 3.10 Main net and Test Nodes

Now you can see that account 1 has been created and it has some sort of address assigned to it. When we create an account in MetaMask, it has three distinct things, Account address, Public key and Private key. These three things make up an account on Ethereum network. Account address is something like a username. We can share it with other people and it's unique in the world. Public and Private key combined make a password, these keys authorize the sending of ether to other accounts. Private keys are not meant to be shared. If shared with someone unknown, they can make use of all the funds in your accounts.

These three pieces of information are stored as Hexadecimal numbers. Hexadecimal numbers are incomprehensively long and impossible to decode. This makes Private key more secure.

One interesting thing about Ethereum is that the account address is distinct and unique across all networks. For example, if you create an account on Gmail and yahoo. Both these accounts have the same username but with different suffixes. But in Ethereum even we if I connect my account to main network or ropsten test network, my account address will remain the same.

So right now my account has zero ether. We can transfer ether into our account by rinkeby faucet. You can access faucet in two different ways. First access this website. <http://www.rinkeby-faucet.com/> You will see an option for giving an address.

Copy the account address from MetaMask and paste it in there.

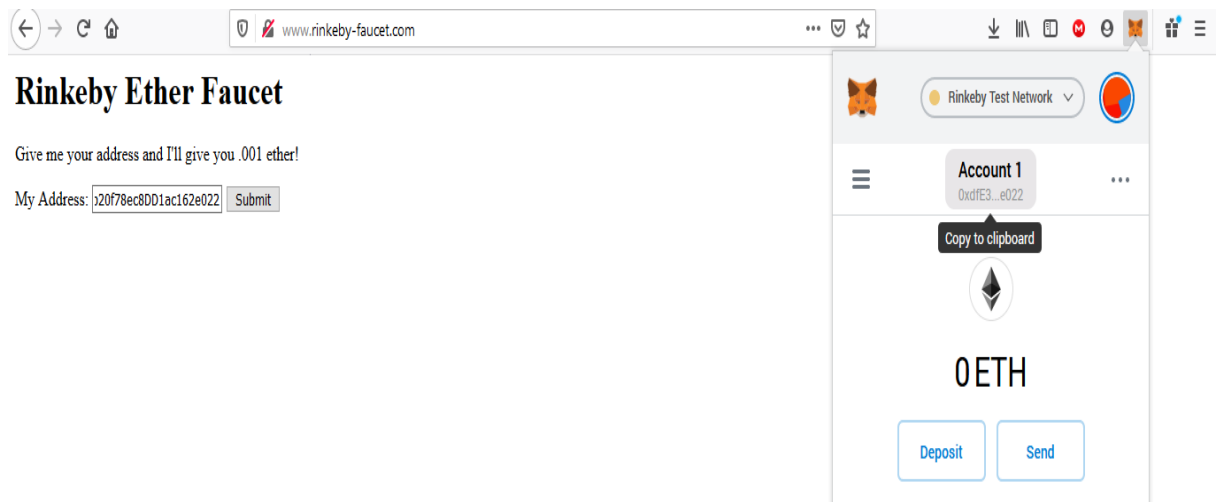


Figure 3.11 Display of how Rinkeby Faucet Works - 1

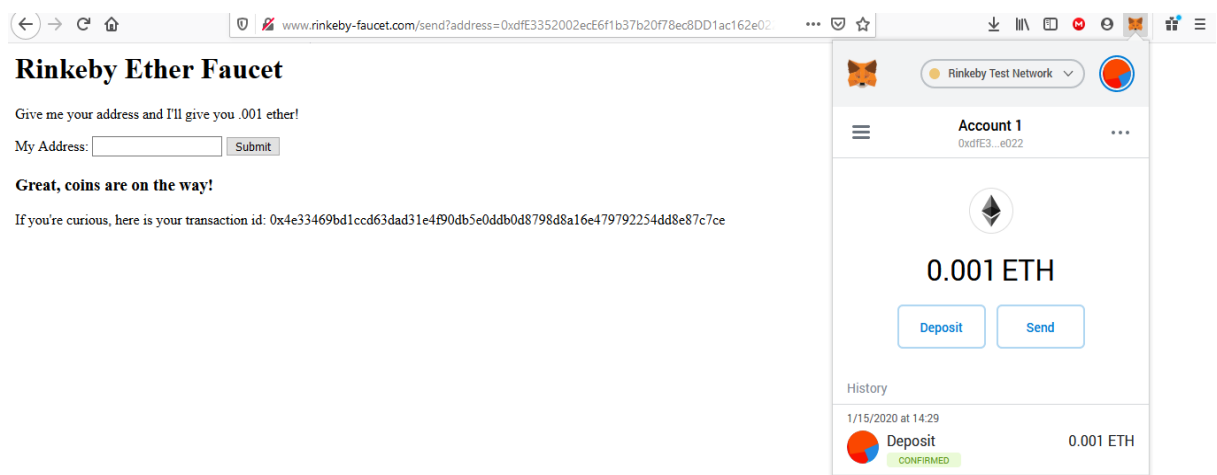


Figure 3.12 Display of how Rinkeby Faucet Works - 2

After submitting the address, you will see loading being happened on faucet page. Transfer doesn't happen simultaneously. After the transfer, you will see the below message and check the balance in the account as well.

Second way to transfer ether is from this site. <https://www.rinkeby.io/#faucet> . You have to request funds via a social networking account. Copy paste the address of the account as a status on your social account. Then copy the link of your status in the faucet site.



Figure 3.13 Twitter Window

As mentioned earlier it has some restrictions on how much ether you can earn in a particular time.

nticated Faucet



Figure 3.14 Rinkeby Faucet

After selecting any desirable option from drop down menu, you can click give me Ether. Again it will take some time for processing. After a while, it will show funds in your account.

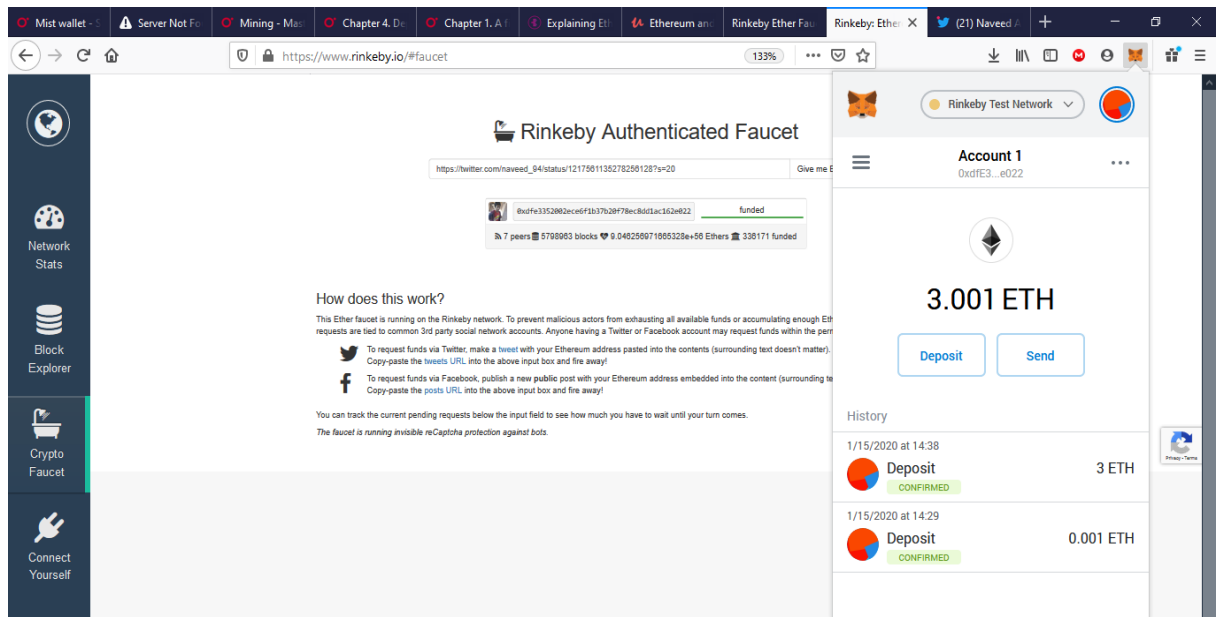


Figure 3.15 Successful Transfer

Now I have 3.001 ETH in our account. It's only connected to rinkeby network. If you select any other test network, you will have zero ethers. Ether mined in Ethereum any test network is decoupled from the other test networks. This ether doesn't have any value in real world currency.

REMIX: -

For the development purposes in Ethereum, developers normally use command line CLI or truffle. But using these sources to deploy smart contracts makes it really complicated to debug and interact with contracts. There is a compiler made specifically for testing and running contracts in solidity language. Remix is a web based tool, so we can access it via web browser. You have to download it on your computer if you want to use it while not being connected to internet. [43]

Remix has a miniature version of Ethereum running in its background. When you compile a source code in remix, solidity compiler will the code in to bytecode. And then that bytecode will be deployed to in browser Ethereum network. Below Image depicts this in clearer way.

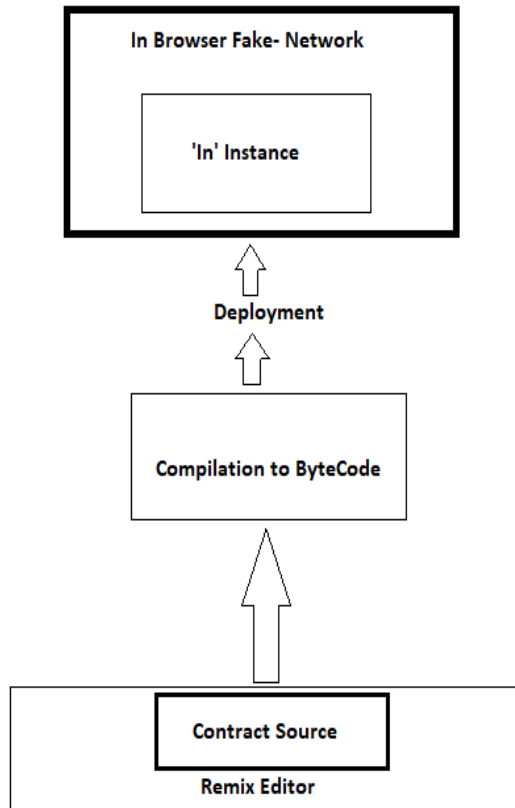


Figure 3.16 Block Diagram of how Remix Works

[45]

Remix is an IDE for solidity. It provides a way to deal with contracts more fluently and has a lot of built in tools for analysing and debugging solidity codes. Remix can be accessed from this site. <http://remix.ethereum.org/>

The first page of remix would look like this:

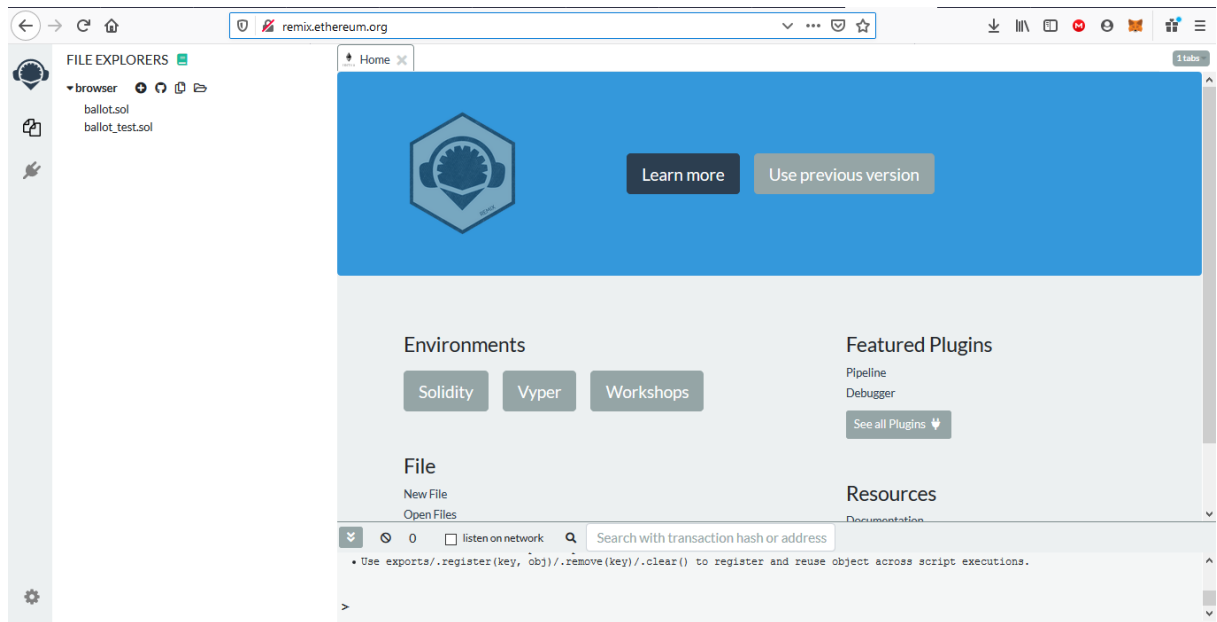


Figure 3.17 Remix Front Page

Remix main page is divided into four parts. Icon panel, Swap Panel, Main panel and Terminal. Codes are written in main panel. Terminal below the main panel shows any errors or warnings in the code executed. It shows all the operations and transactions going on while running the code. On the left of page, you will see swap panel. Swap panel is basically for switching between different codes written at one time. If you click the Icon panel, you will see a list of plugins that can be activated in remix. Below is the list of most common plugins for remix.

Name	Purpose
Compiler	Compiles a contract
Run & Deploy	Sends a transaction
Debugger	Debugs a transaction
Analysis	Presents data of the last compilation

Figure 3.18 List of Common Plugins in Remix

[44]

You have to activate these plugins in order to run the contract.

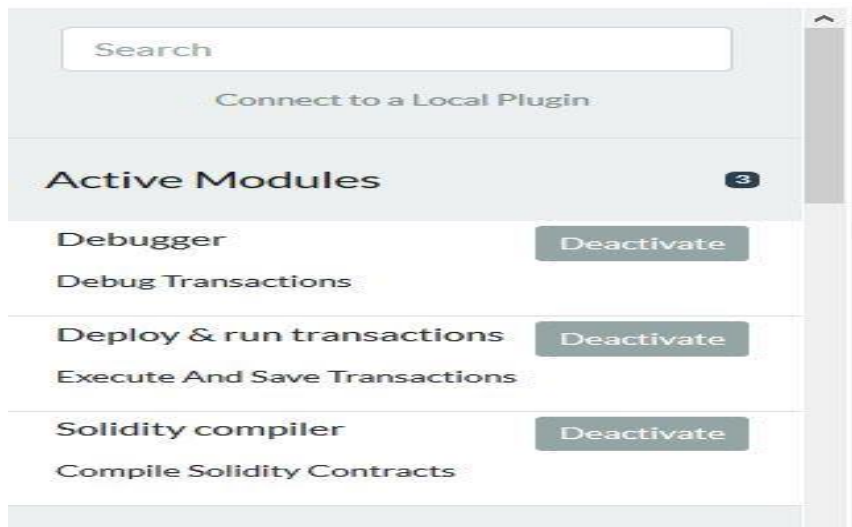


Figure 3.20 Activate Plugins

By default, there is a code for the ballot. You can just remove that and start by writing your own code. Open a new tab and start writing your own code. It would be like the image posted below:

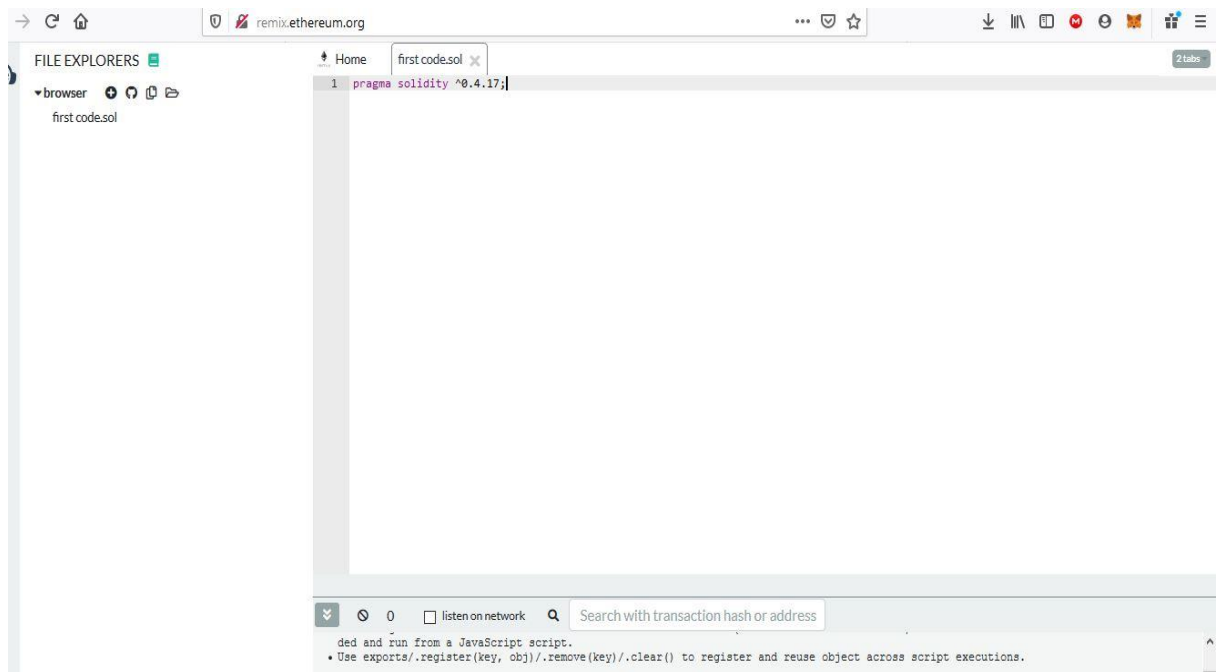


Figure 3.21 Old Remix Front Page

This is an old version of remix editor. I am using an older version for the project.

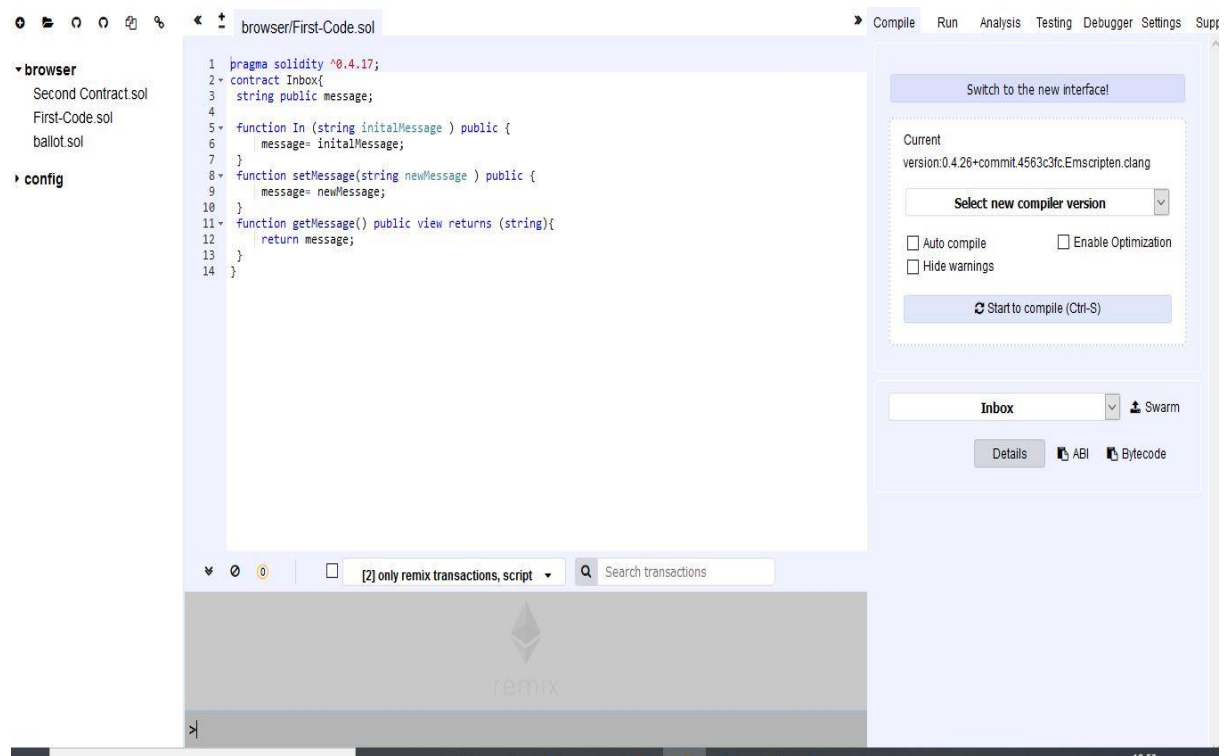


Figure 3.21 Remix Front Page with written code

If you click the arrow on top left side of code written, left window will be minimized. You will be left with three tabs.

Bottom tab is the console tab. Which displays all the things that are happening inside the fake Ethereum network. It will be empty at start because no one has interacted with network.

When you click Run in right most window, you will be given different options. Firstly, which environment you want to run this code into. By default, injected Web3 is selected. We have to select JavaScript VM in order to run the code in network. It will automatically give us some accounts with 100 ethers in each. These accounts only exist in our browser. Value is used to doing a transaction and also sending ether with it.

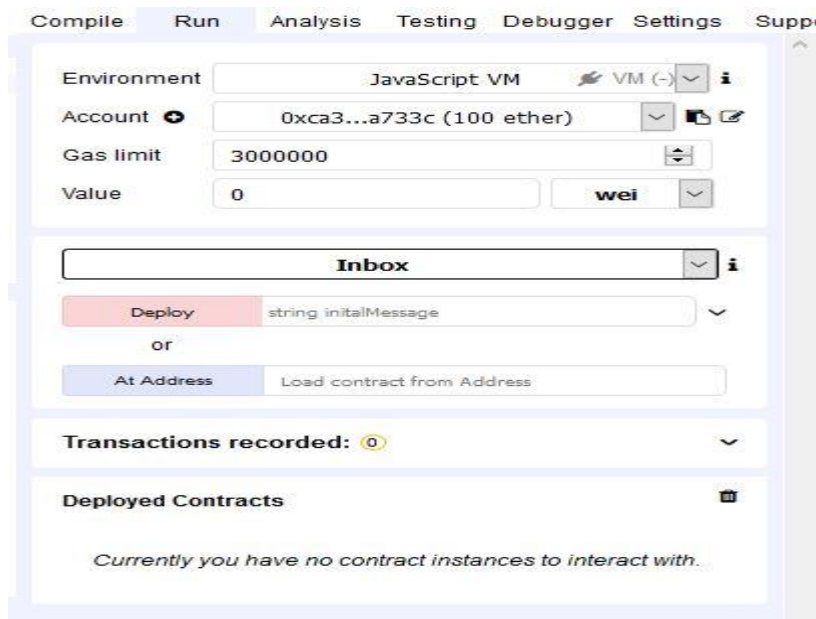


Figure 3.22 Right side of remix

The first code deployed in remix editor. Code will be explained in detail later on.

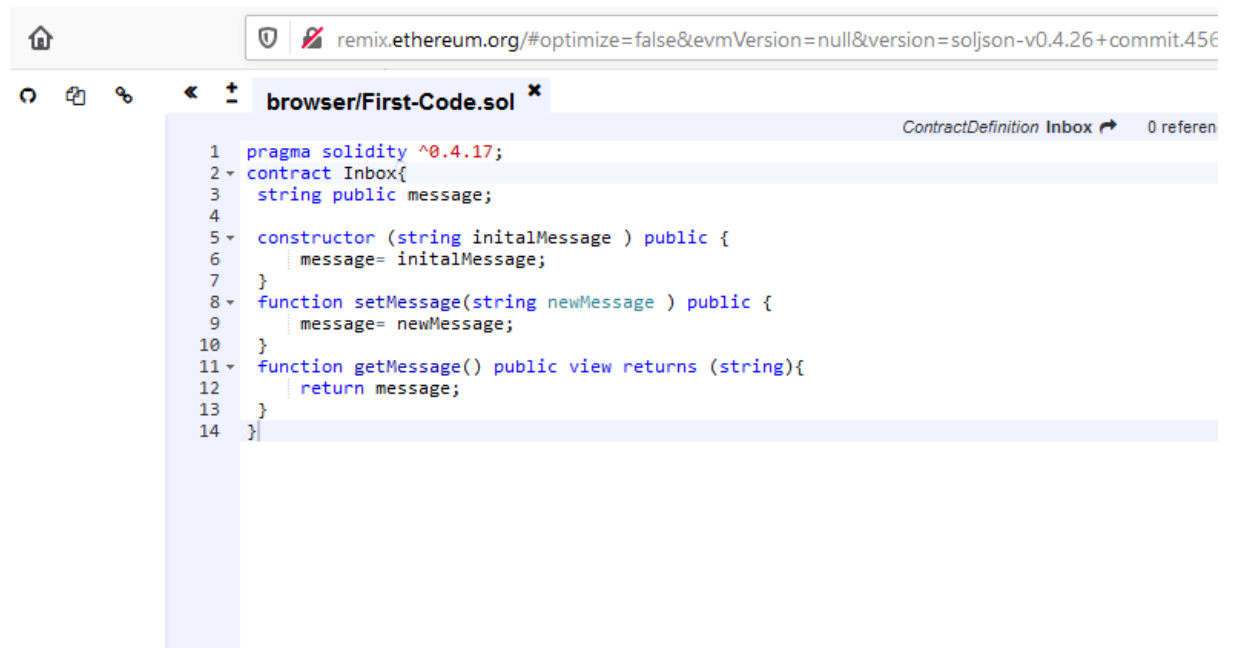


Figure 3.23 Code Written in Remix

If the contract has no syntax or logical errors, you will see a tab on left side for compiling. You should be able to compile the code. Contract name will appear in a box if there are no errors. ABI and Bytecode output of code can be viewed.

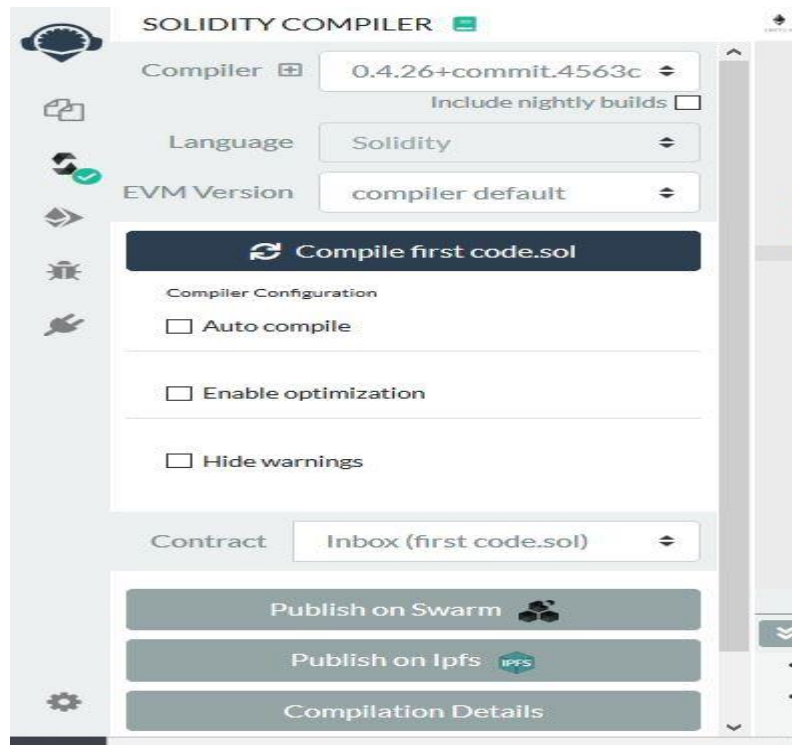


Figure 3.24 Compiling Options

3.3 Demonstration of how Contract Works: -

So, the first line in our code *pragma solidity ^0.4.26*, is used to specify the version of solidity being used in code.

Second line *contract Inbox{* , is to define the contract. Contract name is Inbox. Parentheses will start after this command and everything defined in this code will be in inside these parentheses.

string public message; This command defines a variable message inside the contract. The type of this variable is string and its nature is public. Public means this variable will be accessed from anywhere in the world.

Now we have three different functions associated with this contract. First function is Inbox defined in line 5. The returned argument will be string named InitialMessage. We will make this function public. Inside the function, we have defined that initial Message is equal to Message. Second function defined is setMessage. Argument returned is string called newMessage. Inside this function body we have defined, *message= newMessage;*

Last Function defined is *getMessage*. After defining the last function, we should be able to see a green icon named Inbox on the right side of page. There shouldn't be anything red. If there is anything red, this means there is an error in code. You will see a mark on left side of particular command having an error.

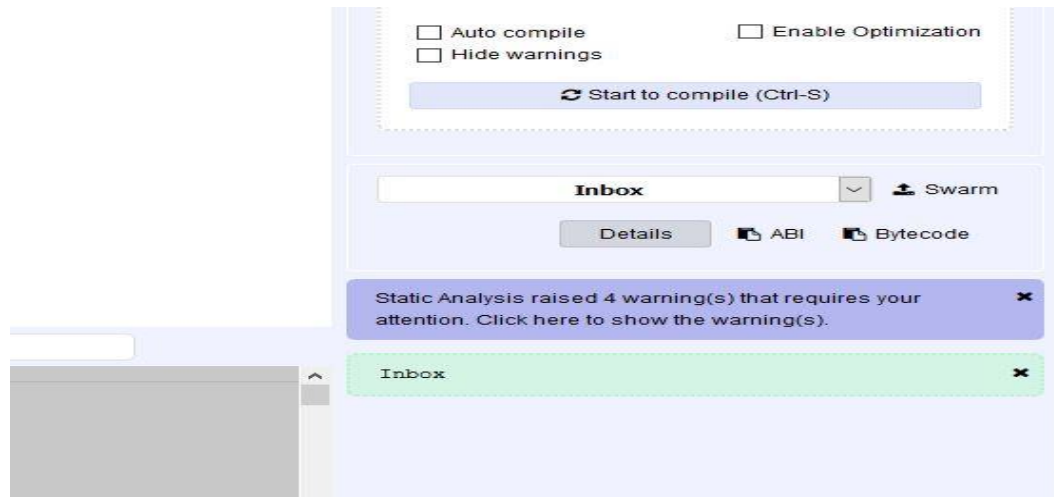


Figure 3.25 Errors after running the code

Every line that has variable declaration or actual line of code should have semi-colon at end, Except the functions.

When you switch to the run tab, you will see drop down menu on which inbox is already selected. This shows the contract which we are trying to run on Ethereum network.

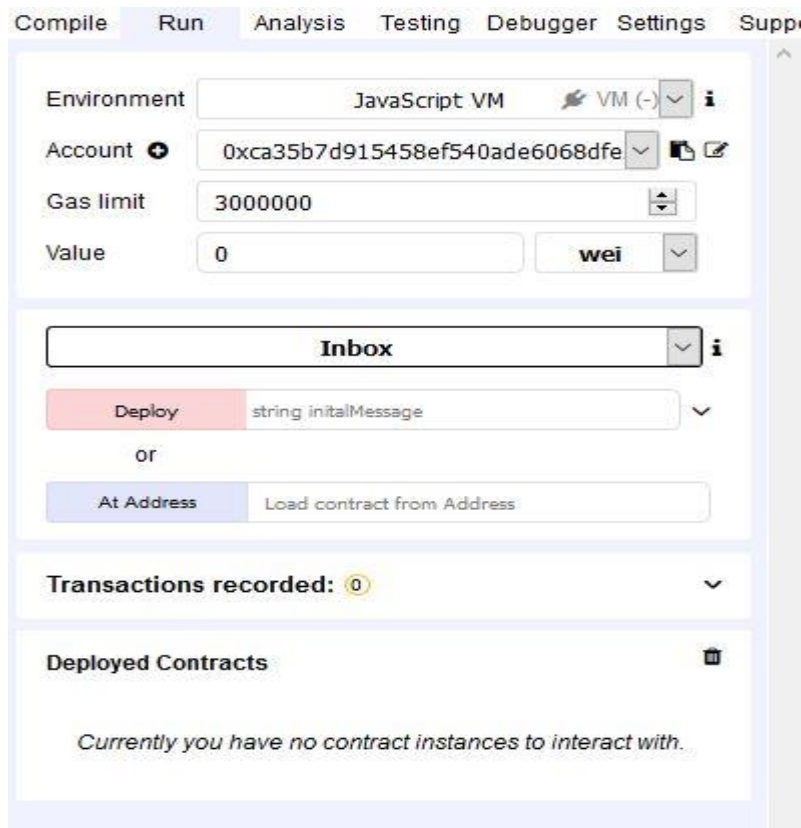


Figure 3.26 Contract Deployed on Blockchain

The first function in our code has the same as the contract name. This is called a constructor function. Constructor functions are automatically called whenever we deploy our contract. So you see a tab of string initialMessage with deploy button next to it. String initialMessage is the argument expected to return form this function.

If we enter any string in tab, constructor function will be called with the message in tab and an instance of contract will exist on the network. I wrote a string 'Hello There'.

Then press the drop down. And click transact.

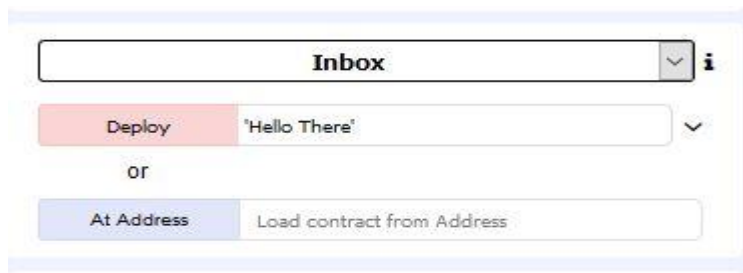




Figure 3.27 Output of Contract

We will see some messages in console at end of page.



Figure 3.28 Console Messages

We will see instance of contract deployed that bottom right side of page. This instance is deployed on network. We can see three different tabs by which we can interact with contract. Tabs which are green are view or constant functions. Which means that they can be called and they will return data at the same moment. Red button shows that we can make changes in contract by calling that function. We can notice that setMessage will also take a value like constructor function. String newMessage indicates that whatever we type there will be given to contract in the exact same way.

We can see in the second image that when we press getMessage button, we will be returned with the zero string. This means the first of one value which will be returned.

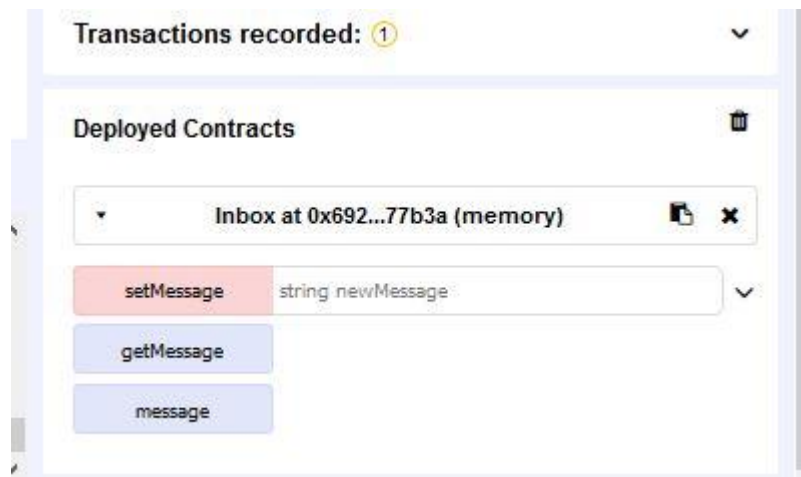


Figure 3.29 Experimenting with code - 1

If in code we have specified to return multiple values, we should see zero, one and two and so on. After that, there is the mentioned string. This means the type of data we have specified in code to be returned. At the end, it tells us the actual value of string 'Hello there'.

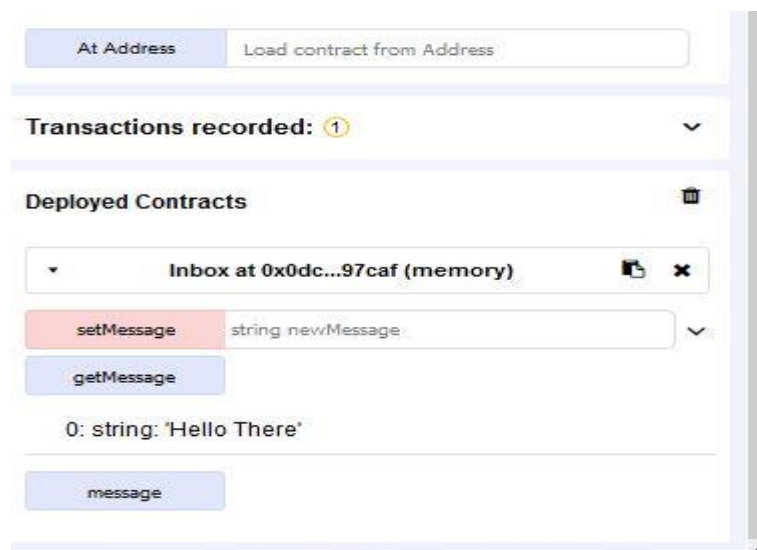


Figure 3.30 Experimenting with Code - 2

So when we type in the tab of setMessage, 'Babye'. We will see some console messages displaying that we are interacting with the contract.

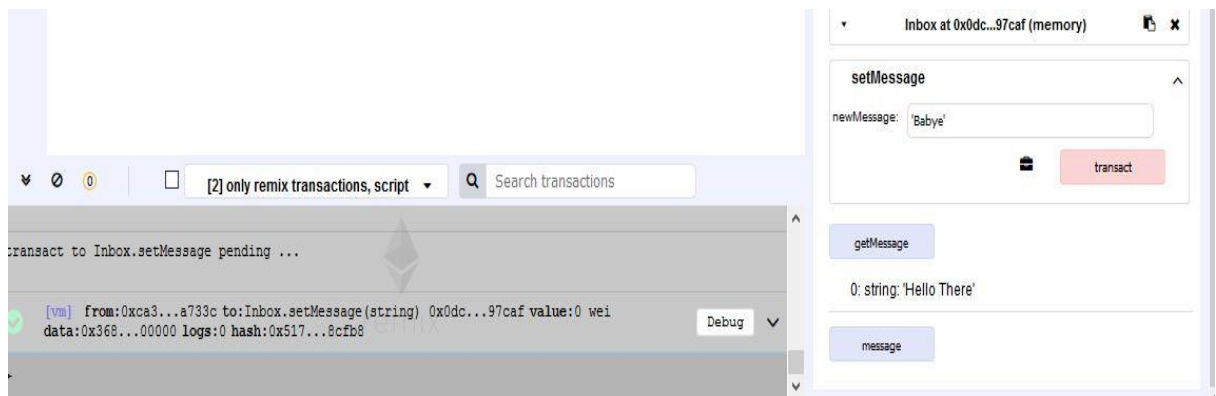


Figure 3.31 Experimenting with Code - 3

By clicking transact, we sending a function to deployed contract and this will invoke the setMessage function in our contract. This will update the value of message variable. You will see that value of getMessage won't be changed automatically. So when you click getMessage, its value will be updated.

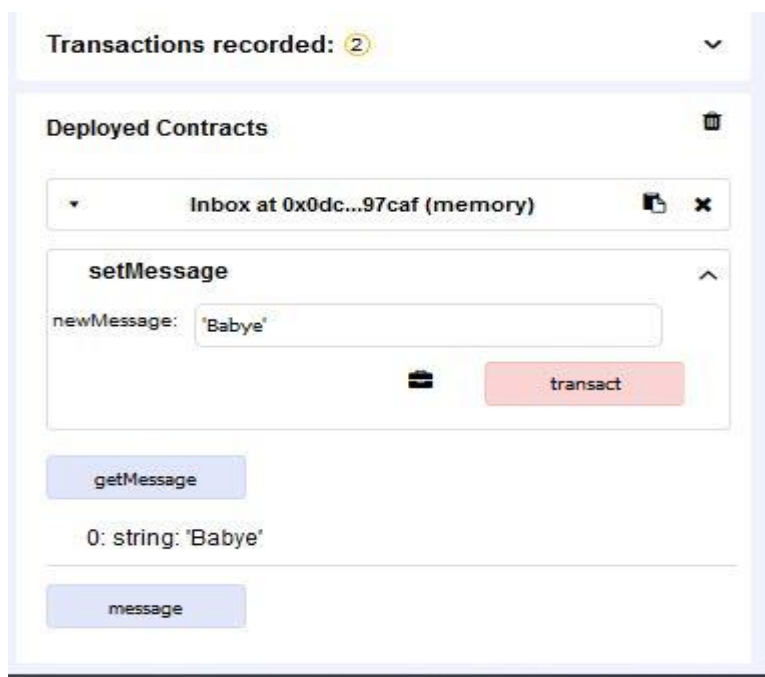


Figure 3.32 Experimenting with code - 4

When we click the third tab written 'message', this will return the same value as getMessage function. In solidity, when we define a storage variable and make that variable as public, the contract will create the duplicate function in the background. Now if we call this function, it will return the variable. This means the getMessage can be

avoided if we use the storage variable as public, because we will a variable created in background.



Figure 3.33 Experimenting with Code - 5

So if we modify our code and delete the getMessage function, we will still see the same output.

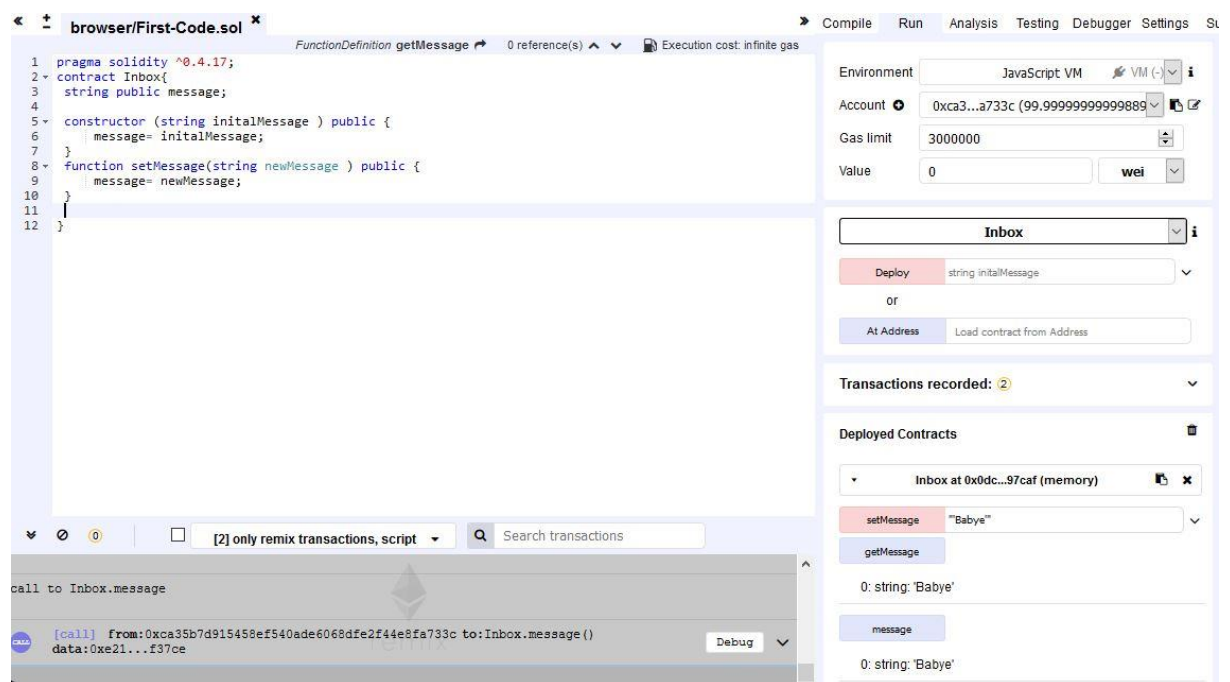


Figure 3.34 Experimenting with code - 6

There is an instance of an old contract in bottom right side of editor. We can cancel it by clicking on the cross. Then we will compile the new code.

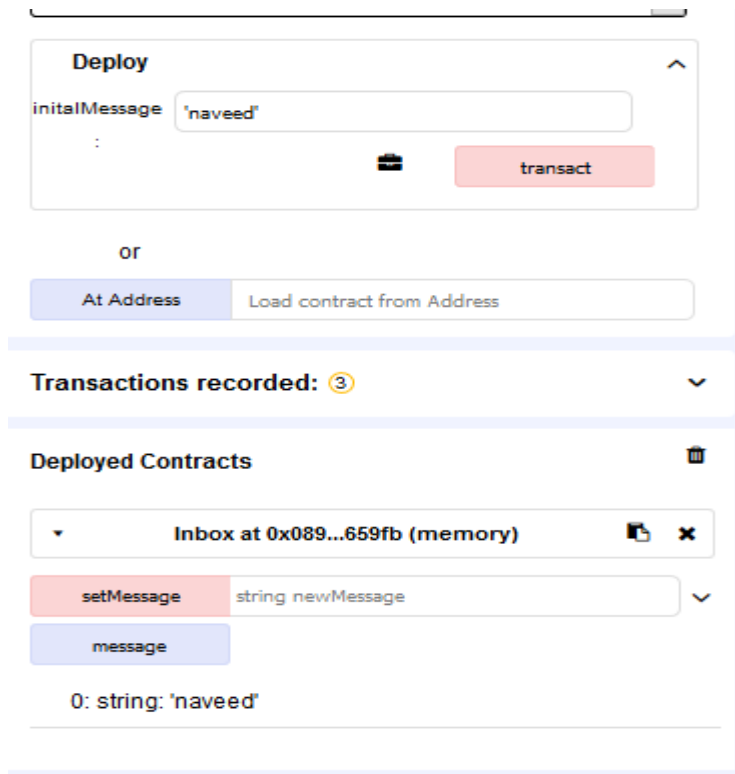


Figure 3.35 Experimenting with Code - 7

So now there are two tabs instead of one. I set the initialMessage as 'navid'. And clicked transact. I clicked message tab, and will see the value of message variable as set above.

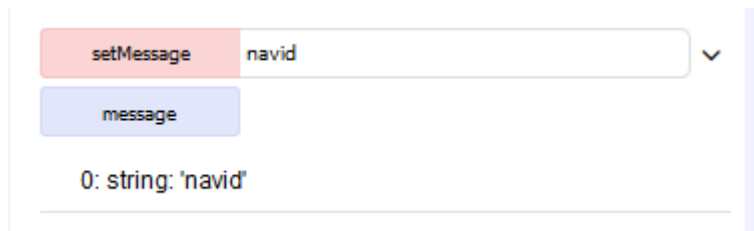


Figure 3.36 Experimenting with code - 8

Set the value of setMessage function and click that tab. Afterward click message and will see the same string of message.

Second Code: -

This code is explained in Geth lab.

```

1  pragma solidity ^0.4.17;
2  contract SimpleCoin {
3      mapping (address => uint256) public coinBalance;
4      constructor() public {
5          coinBalance[0x14723A09ACff6D2A60DcdF7aA4AFF308FDDC160C] = 10000;
6      }
7      function transfer(address _to, uint256 _amount) public {
8          coinBalance[msg.sender] -= _amount;
9          coinBalance[_to] += _amount;
10     }
11 }
12

```

Figure 3.37 Second Code

After running the remix compiler in browser, you will be asked to connect that with metamask extension. A message like the below will appear in your browser.

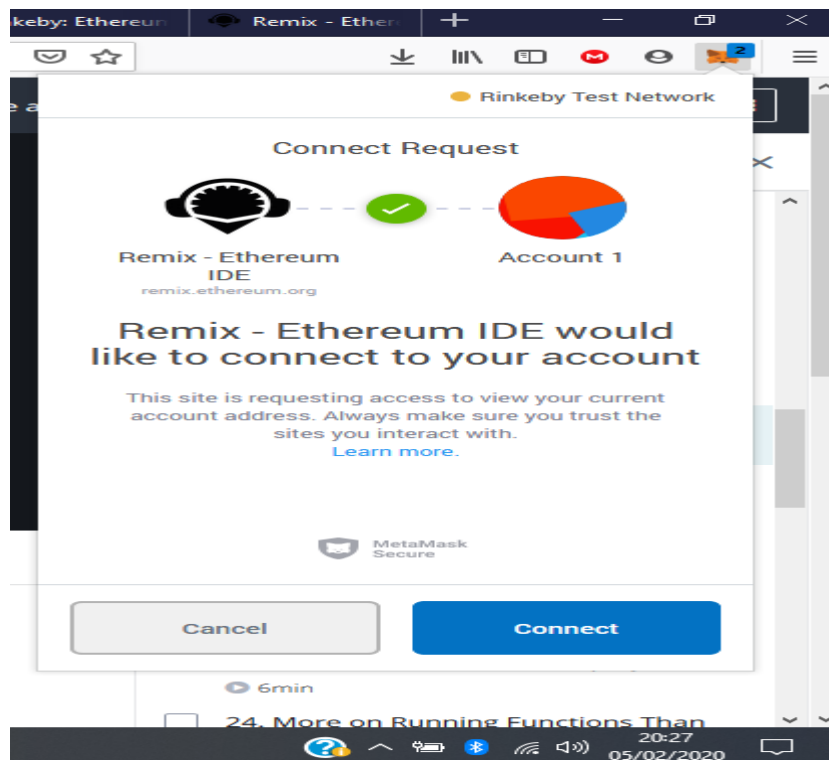


Figure 3.38 MetaMask Display Message - 1

After you compile this code, you should go to run tab. You will see deploy and At address tab.



Figure 3.39 Output of code

This contract will automatically connect to your metamask account. Account address will be your account address in metamask. When you deploy this contract in Ethereum Blockchain, you will be asked to confirm this change.

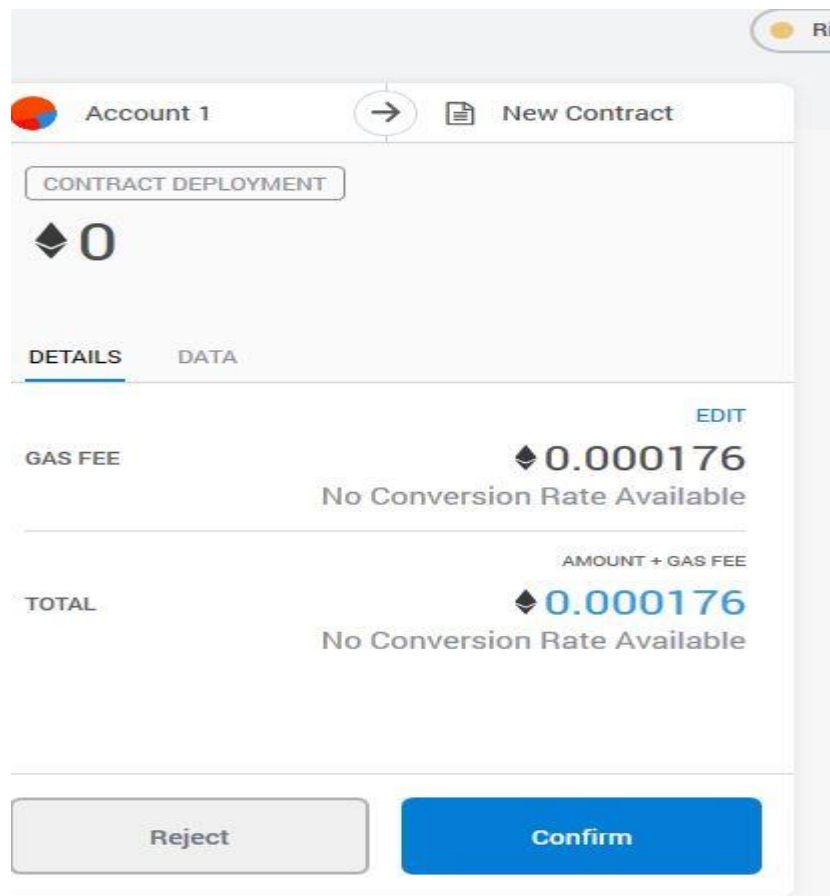


Figure 3.40 MetaMask Display Message - 2

You will see the deployed contract notification in remix as well. It will be saved on Blockchain.



Figure 3.41 Tabs after compiling code

You can see in the above output that there are two tabs, red and blue. Blue buttons perform the read operations against the contract while red button performs write operations. Read operations are such as to check the value of state variables and write operations call the functions to change the value of state variables.

In this code coinBalance is blue because it will read the coin balance against associated address. While the transfer is red because it will write or change the value of the variable.

I have created two accounts. Balance in my first account is 3.001 ether and zero ether in second one.

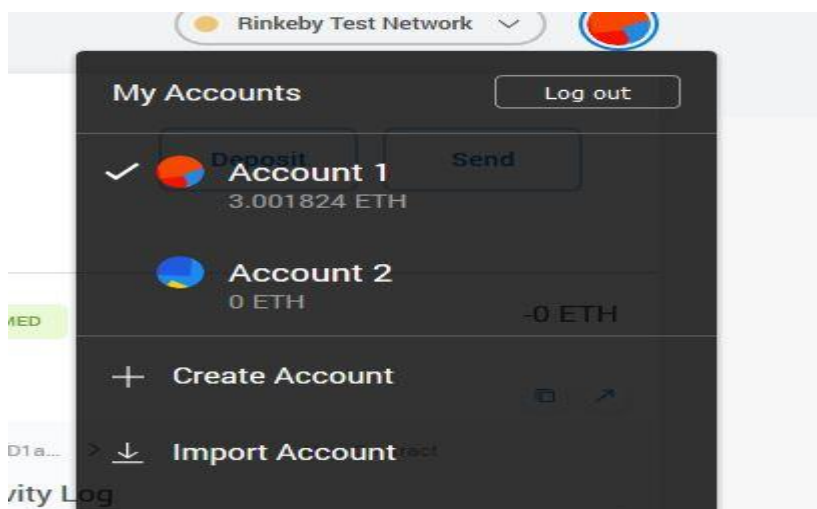


Figure 3.42 MetaMask display Message - 3

0xdfE3352002ecE6f1b37b20f78ec8DD1ac162e022 First account address.

0x4fc5B068bbEA2c2e2EBD8cFD0aEFda3a830ef5A8 Second account address.

When you will deploy this contract against my first account. A notification will appear whether to confirm or reject the transaction in metamask. If you approve it, you will see confirmation message in metamask. It will contain all the details of transaction and its reaction on ether scan.

The screenshot shows a transaction history entry for a contract deployment. The transaction is confirmed and has a value of 0 ETH. The details section shows the transaction was created at 21:43, submitted at 21:44, and confirmed at 21:44 on 2/5/2020. The activity log shows the transaction was created with a value of 0 ETH, submitted with a gas fee of 0 WEI, and confirmed at 21:44 on 2/5/2020. The transaction details table shows the amount is 0 ETH, gas limit is 175753, gas used is 175753, gas price is 1 GWEI, and the total is 0.000176 ETH.

Transaction	
Amount	0 ETH
Gas Limit (Units)	175753
Gas Used (Units)	175753
Gas Price (GWEI)	1
Total	0.000176 ETH

Figure 3.43 EtherScan - 1

Ether scan output: -

The screenshot shows the overview of a transaction on EtherScan. The transaction is successful and has 2 block confirmations. The transaction hash is 0x086755e123301b8db9f3a496ef7a494a0a32085ed42fe77a5e7fade5a639642d. The transaction was created 16 seconds ago (Feb-06-2020 04:44:41 AM +UTC). The transaction was sent from 0xdfE3352002ecE6f1b37b20f78ec8dd1ac162e022 to a contract 0x26f04e2f3a4ceef43932ed6a9b42abf0cfd86965. The value is 0 Ether (\$0.00).

Transaction Hash:	0x086755e123301b8db9f3a496ef7a494a0a32085ed42fe77a5e7fade5a639642d
Status:	Success
Block:	5921613 2 Block Confirmations
Timestamp:	16 secs ago (Feb-06-2020 04:44:41 AM +UTC)
From:	0xdfE3352002ecE6f1b37b20f78ec8dd1ac162e022
To:	[Contract 0x26f04e2f3a4ceef43932ed6a9b42abf0cfd86965 Created]
Value:	0 Ether (\$0.00)

Figure 3.44 EtherScan - 2

So when you want to send ether from one account to another, you can simply click on deposit.

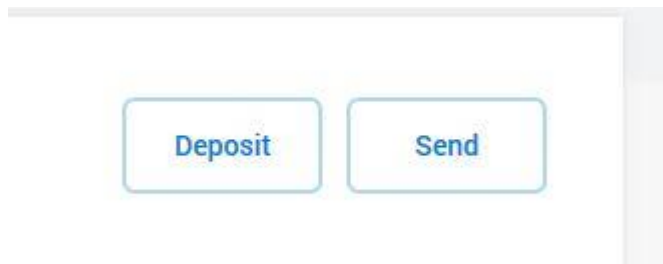


Figure 3.45 MetaMask Display Message - 4

You will be asked to enter the recipient account's address. You can set the transaction fees as per the time requirement for transaction.

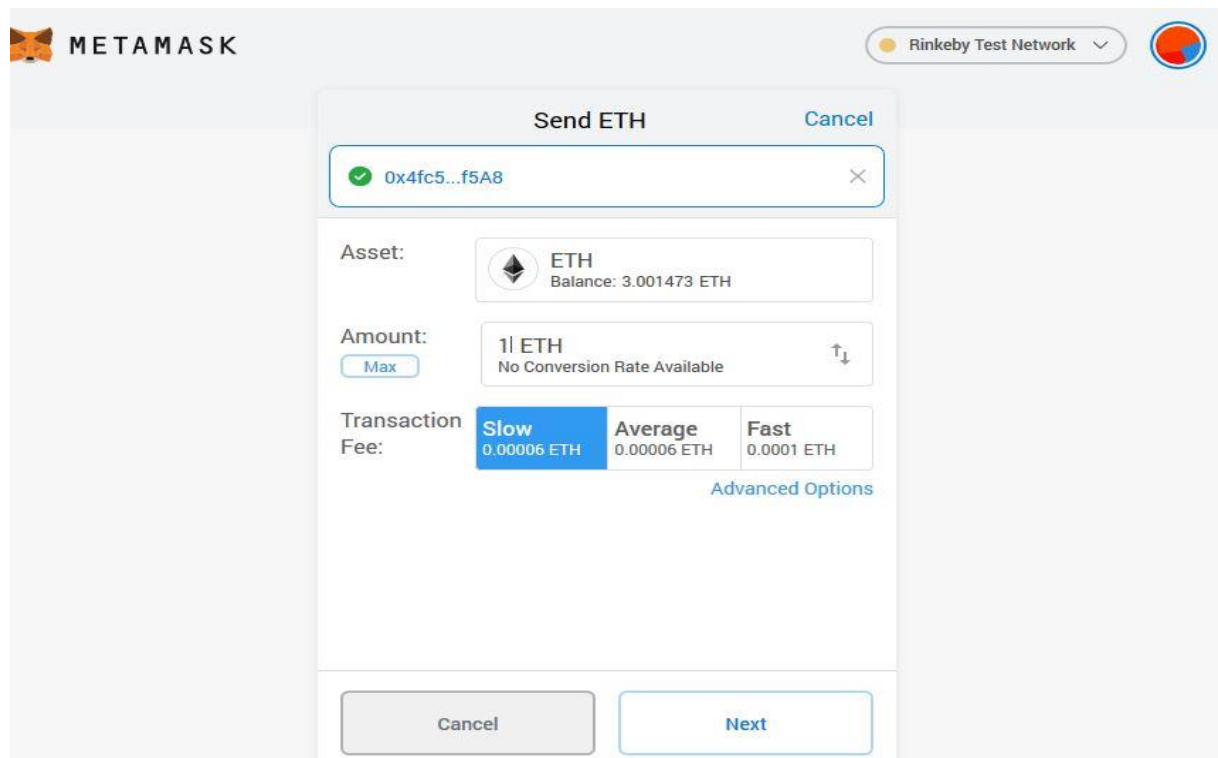


Figure 3.46 MetaMask Display Message - 5

After that you will be shown gas fee and total amount of ether it's going to cost you by doing this transaction.

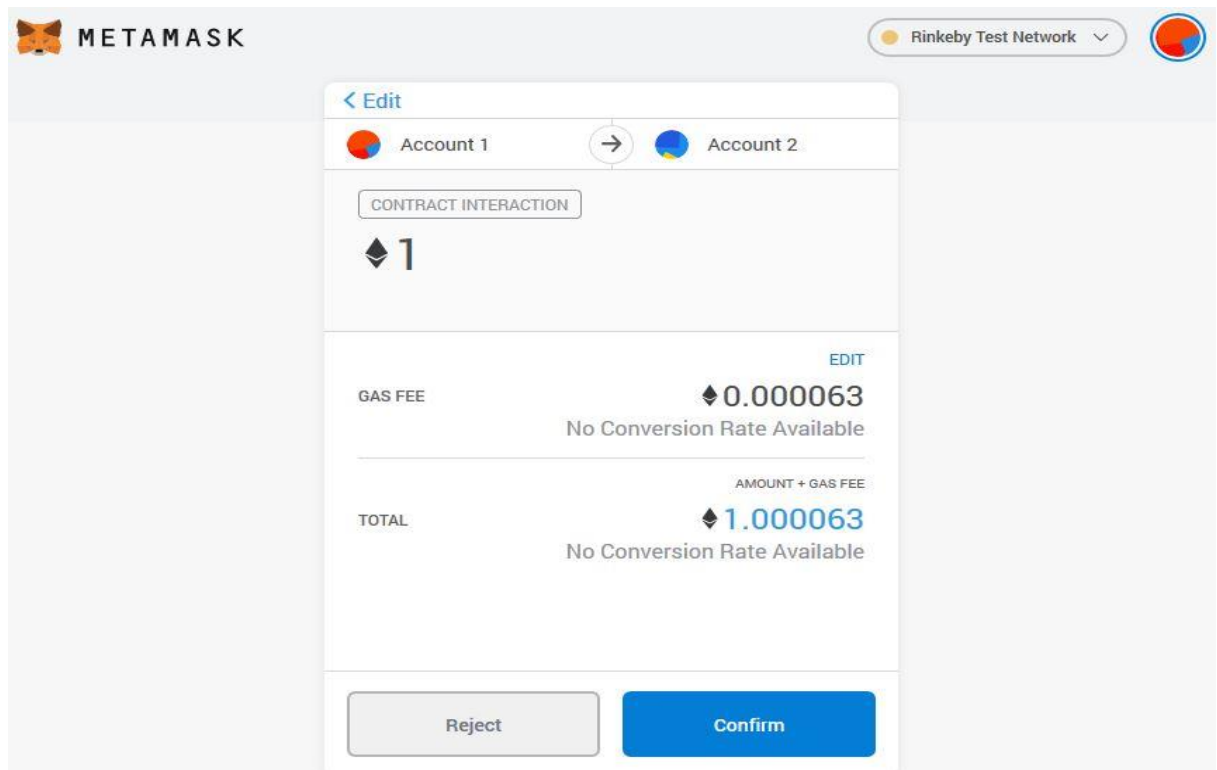


Figure 3.47 MetaMask Display Message - 6

After you confirm, it will take some time to process the transaction. Again you can view this on ether scan.

I transferred one ether from account one to account two. Below are the confirmation outputs.

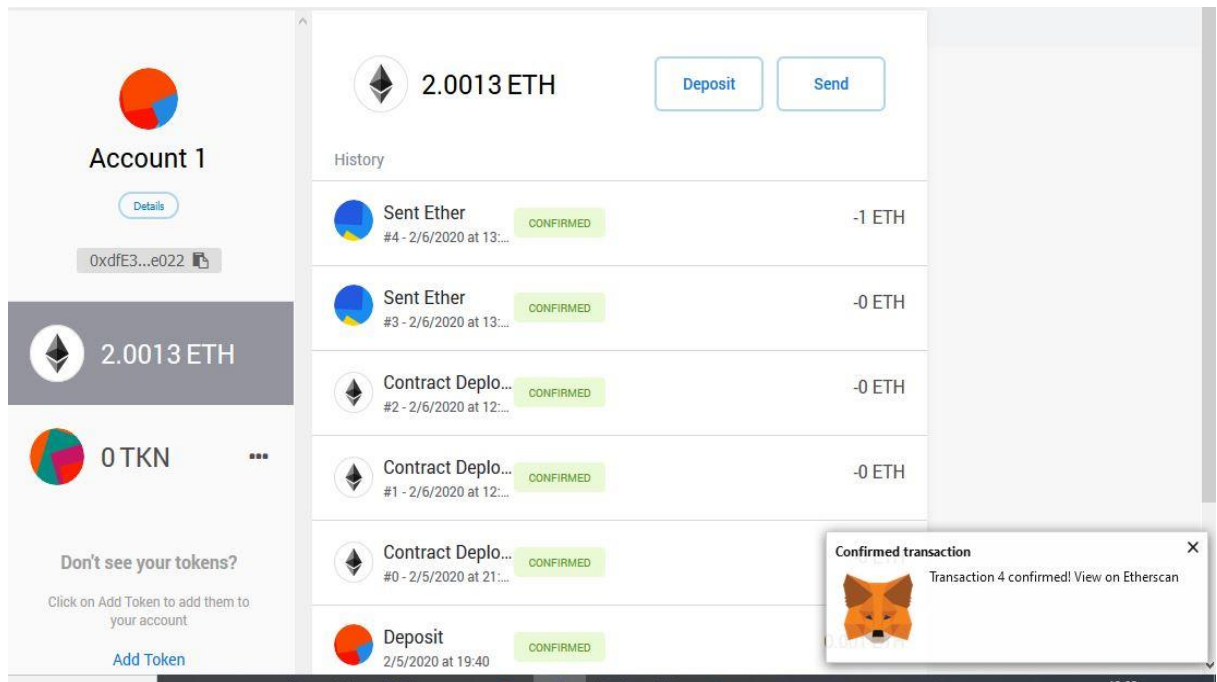


Figure 3.48 MetaMask Display Message - 7

You can see the decrement in account balance of account 1 to 2.0013 ether. Account 2 balance is increased from zero to one ether.

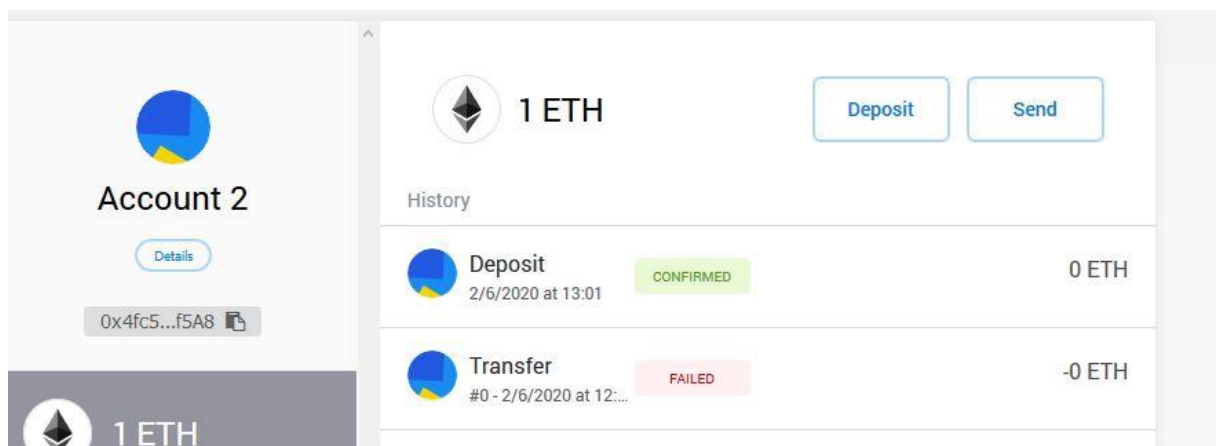


Figure 3.49 MetaMask Display Message - 8

Details of transaction in account 1.

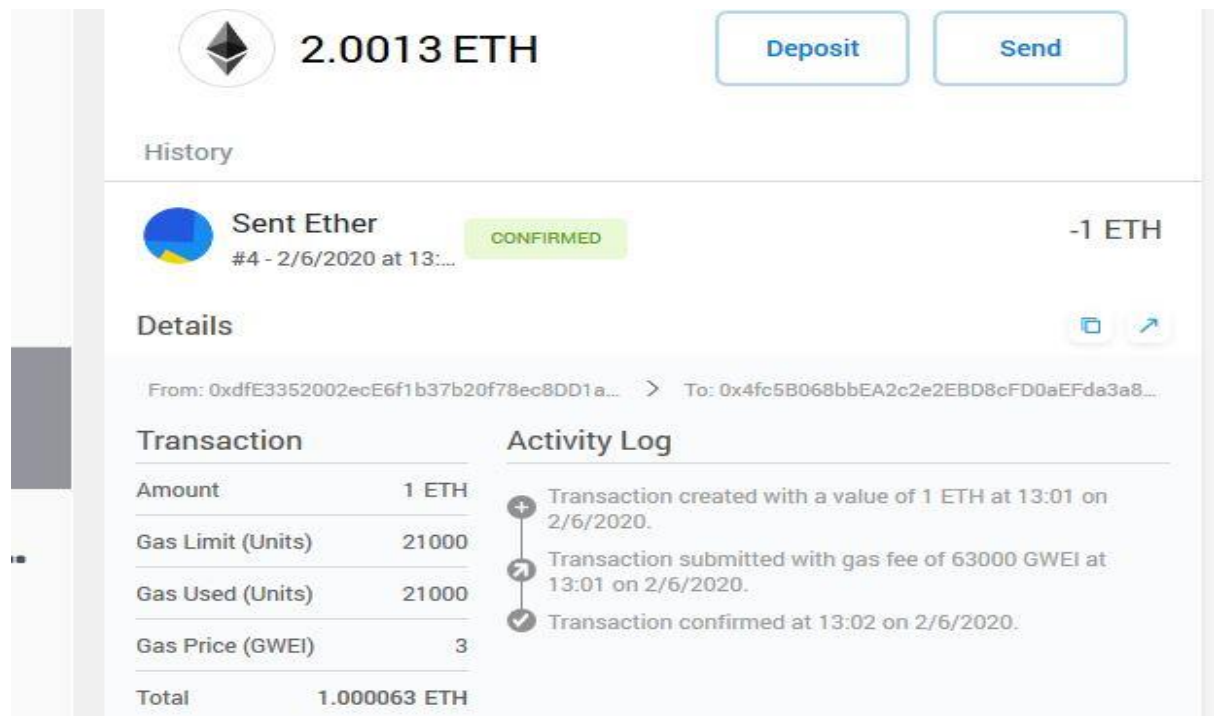


Figure 3.50 MetaMask Display Message - 9

*This code and its demonstration is partially taken from Roberto Infante book ‘Building Ethereum Dapps’

3.4 Explanation of Code: -

The screenshot shows a browser window with the URL `remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.4.26+commit.456`. The browser tab is titled 'browser/First-Code.sol'. The code editor displays the following Solidity code:

```

1 pragma solidity ^0.4.17;
2 contract Inbox{
3   string public message;
4
5   constructor (string initialMessage ) public {
6     message= initialMessage;
7   }
8   function setMessage(string newMessage ) public {
9     message= newMessage;
10  }
11  function getMessage() public view returns (string){
12    return message;
13  }
14 }

```

Figure 3.51 Code Written

[45]

First line in our contract specifies the version of solidity being used. This helps the solidity compiler to customize the way to show outputs of contents in code as per mentioned version. This is done to ensure that when solidity is upgraded or new syntax is added in language, all the old contracts will still be working because the compiler will see the version specified on top of code and compile it accordingly. So the new features in solidity will automatically be ignored.

In second line, we make our first contract definition. Word ‘contract’ is a key word, it is almost identical to class in Java language. All the contracts deployed in Ethereum network can be thought as classes. These deployed versions of classes are instances in Ethereum network. So in second line, we tell the compiler that this is a contract and its name is Inbox.

Directly after defining contract in line three, we define a storage variable. These storage variables act as instance variables in Ethereum, that exists for the life of contract. String is the type of variable being defined. It means that variable message will only contain string of data. There will no other types of data such as array or numbers. Public keyword defines who can access the contents of this contract. Third word in this command is message which specifies the name of this storage variable. Value of storage variable is stored and changed automatically in Ethereum Blockchain. So any change in the value of message will result in the change of storage variable. In contrast to storage variable, local variables are created once in the life of contract and discard after being used for one time.

After defining the storage variable, we have defined three functions. All three functions defined will be called after running the contract in Blockchain.

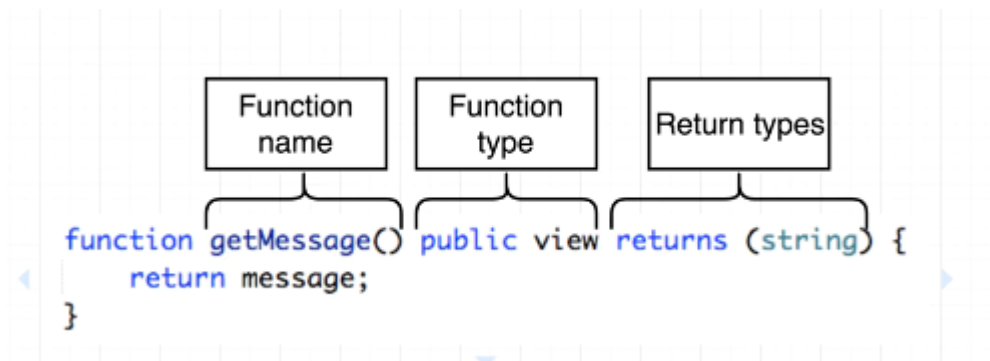


Figure 3.52 Function Explained

Moving on to defining functions, we will see line 11 of the code. We have defined getMessage function. First part of command contains function keyword. getMessage is the name of function. There is no given format in solidity language for picking a name of function. getMessage specifies that this functions will get and retrieve message from contract. Set of parenthesis at the of getMessage indicates the type of argument this function can take. In this particular command, getMessage will not take any kind of argument.

After the function name, we have function type. Function type could have different syntax and different wordings as per the contract. Most common function types in solidity are shown in table below.

Public	Anyone can call this function
Private	Only the specified contract can call
View	Function returns data and doesn't modify the contract's data
Constant	Same as View function
Pure	Function will not even modify or even read the contract's data
Payable	When someone calls this function, they have to send ether with it.

Table 5 Common Function types in solidity

By making the function public, anyone with the account on Ethereum Blockchain will have access to contract and can call the given function. Public functions can be a security issue, since anyone can view the data in them. As opposed to public, private function returns data that is only available to our contract's code. Making a function private doesn't implement security. Private function acts like a helper function. Private function is only specified for that contract, if any engineer wants to get data from that contract he would have to call this specific function. You can make a function either public or private.

View and constant functions do exactly same kind of work. Constant was used in older versions of solidity while now it is being replaced by view in newer versions. View and constant can be used interchangeably. We are using view and constant while declaring functions specifies whether a function modifies data or not. In our code, function `getMessage` uses view keyword. This is because we are not going to change or modify data, we are using it to only access data.

After constant, we have a function type called pure. Pure keyword doesn't change or modify the data and it don't even access the data. Pure is not much used in contracts. Last keyword is payable. This keyword means that if someone from outside the calls the function within the contract and send money to the contract at the same time.

In line eight, we have declared a function called `setMessage`. In this function, we have set a requirement for an argument return that it should be in string format. `setMessage` function is not view or constant because we are changing the data within our contract. We are changing the variable message.

Returns keyword is used to specify the type of return value that we can expect from function. Return can only be used with view and constant keyword because we are not trying to modify to change any data.

Below diagram depicts how this contract works. We have deployed a contract Inbox on fake Ethereum network using remix editor. This contract has a variable called message which is in storage of contract. Message variable can either be modified or changed by `getMessage` or `setMessage` function. `getMessage` is used for fetching the value from contract and `setMessage` is used to set the value in contract. Both of these functions can

be called by anyone on the network. So in the end, our contract is storing a message on network which someone else can read.

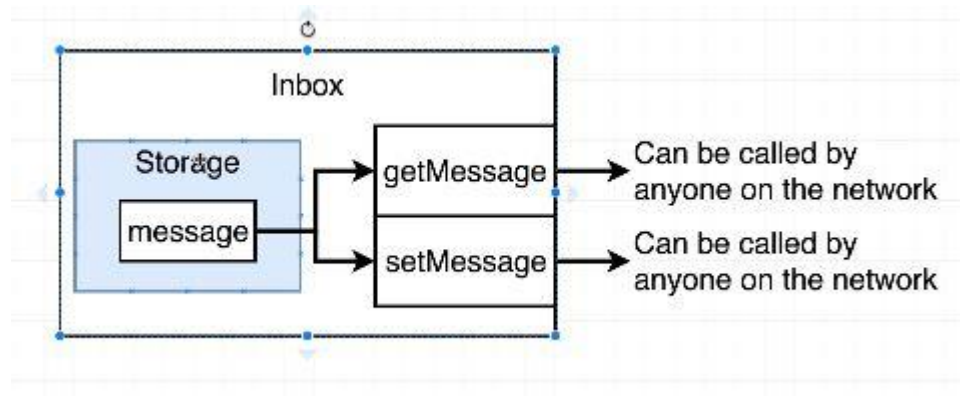


Figure 3.53 Working of our Contract

[46]

3.5 Geth (Go Ethereum) :-

It's a very popular client for Ethereum. Geth is a program which makes your device serve as a node in Ethereum Blockchain. You can use geth to mine, transfer ether and run different contracts on Ethereum virtual machine. Geth can be used to search block history. It can be used to create and manage an account in Ethereum Network. We can develop decentralized apps and run it on Ethereum using geth. Ethereum nodes and clients can be implemented in different languages like Go, C++, Python. Geth uses Go language. It is a versatile command line tool by which we can connect to mainnet and all other test Blockchains. Sometimes, geth is regarded as complicated to operate, so we can use wallets like Mist which use geth in background.

Geth can be installed in windows, Linux and Mac operating system. We can download free and compatible geth versions from this site. <https://geth.ethereum.org/downloads/>

There are several ways to communicate with geth node. Most common are by console and script mode using JavaScript runtime environment, HTTP and IPC inter process communication. If you want to communicate with geth client with HTTP or IPC, you have to use JSON-RPC.

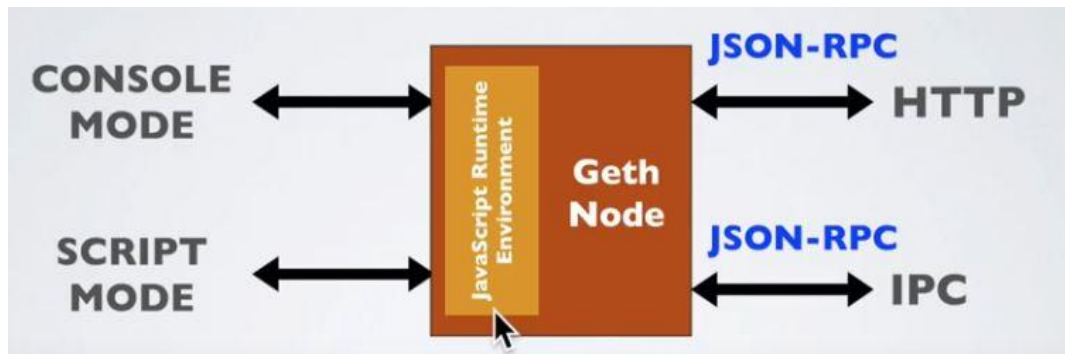


Figure 3.54 Geth Communication Types

[41]

Most common console commands used in geth are personal, eth, web3, admin, debug and miner. If you just type in these commands, it will show a lot off information. You will have to more specific in commands if you want to see precise details.

3.6 Geth Lab: -

I have used Linux based Ubuntu operating system to install geth node. You can download Ubuntu image free of cost from Ubuntu website. Run it on oracle virtual box. Assign the Ubuntu VM most compatible parameters like processing power.

Install geth by running following commands on Ubuntu command line.

```
File Edit View Search Terminal Help
naveed@naveed-VirtualBox:~$ sudo add-apt-repository -y ppa:ethereum/ethereum
[sudo] password for naveed:
Hit:1 http://ca.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://ca.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:3 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:4 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic InRelease [15.4 kB]
Get:5 http://ca.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:6 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic/main amd64 Packages [3,452 B]
Get:7 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic/main i386 Packages [3,456 B]
Get:8 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic/main Translation-en [936 B]
Fetched 275 kB in 3s (102 kB/s)
```

Figure 3.55 Installing Geth on Ubuntu - 1

```

naveed@naveed-VirtualBox:~$ sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 http://ca.archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic InRelease
Get:4 http://ca.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:5 http://ca.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Fetched 252 kB in 1s (214 kB/s)
Reading package lists... Done
naveed@naveed-VirtualBox:~$ sudo apt-get install ethereum
Reading package lists... Done

```

Figure 3.56 Installing Geth on Ubuntu - 2

You can find these commands and instructions on geth website. PPA is for personal package archive. <https://geth.ethereum.org/docs/install-and-build/installing-geth#install-on-ubuntu-via-ppas>

You can run geth by typing in 'geth'. If you run geth with no parameters, it will start synchronizing with mainnet. Mainnet contains all the Blockchain. It will take from days to weeks depending on internet connection and RAM of node.

```

naveed@naveed-VirtualBox:~$ geth
INFO [01-12|14:58:57.904] Bumping default cache on mainnet      provided=1024 updated=4096
WARN [01-12|14:58:57.904] Sanitizing cache to Go's GC limits    provided=4096 updated=1599
INFO [01-12|14:58:57.906] Maximum peer count                    ETH=50 LES=0 total=50
INFO [01-12|14:58:57.906] Smartcard socket not found, disabling err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [01-12|14:58:57.907] Starting peer-to-peer node           instance=Geth/v1.9.9-stable-01744997/linux-amd64/go1.13.4
INFO [01-12|14:58:57.907] Allocated trie memory caches         clean=399.00MiB dirty=399.00MiB
INFO [01-12|14:58:57.907] Allocated cache and file handles     database=/home/naveed/.ethereum/geth/chaindata cache=799.00MiB handles=2048
INFO [01-12|14:58:57.967] Opened ancient database              database=/home/naveed/.ethereum/geth/chaindata/ancient
INFO [01-12|14:58:57.967] Writing default main-net genesis block
INFO [01-12|14:58:58.436] Persisted trie from memory database  nodes=12356 size=1.79MiB time=111.070378ms gcnodes=0 gcsz=0.00B gctime=0s
livenodes=1 liveness=0.00B
INFO [01-12|14:58:58.436] Initialised chain configuration       config="{ChainID: 1 Homestead: 1150000 DAO: 1920000 DAOSupport: true EIP150
: 2463000 EIP155: 2675000 EIP158: 2675000 Byzantium: 4370000 Constantinople: 7280000 Petersburg: 7280000 Istanbul: 9069000, Muir Glacier: 9200
000, Engine: ethash}"
INFO [01-12|14:58:58.436] Disk storage enabled for ethash caches dir=/home/naveed/.ethereum/geth/ethash count=3
INFO [01-12|14:58:58.436] Disk storage enabled for ethash DAGs  dir=/home/naveed/.ethash count=2
INFO [01-12|14:58:58.436] Initialising Ethereum protocol       versions="[64 63]" network=1 dbversion=<nil>

```

Figure 3.57 Connecting to Main Blockchain

We can start using geth console by typing in 'geth console'. You will see a message indicating you have entered console mode.

```

naveed@naveed-VirtualBox:~$ geth console
INFO [01-12|15:07:19.334] Bumping default cache on mainnet
WARN [01-12|15:07:19.335] Sanitizing cache to Go's GC limits
INFO [01-12|15:07:19.336] Maximum peer count
INFO [01-12|15:07:19.336] Smartcard socket not found, disabling
INFO [01-12|15:07:19.337] Starting peer-to-peer node
INFO [01-12|15:07:19.337] Allocated trie memory caches
INFO [01-12|15:07:19.337] Allocated cache and file handles
INFO [01-12|15:07:20.003] Opened ancient database
INFO [01-12|15:07:20.004] Initialised chain configuration
: 2463000 EIP155: 2675000 EIP158: 2675000 Byzantium: 4370000 Constanti
000, Engine: ethash}"
INFO [01-12|15:07:20.004] Disk storage enabled for ethash caches
INFO [01-12|15:07:20.005] Disk storage enabled for ethash DAGs
INFO [01-12|15:07:20.005] Initialising Ethereum protocol
INFO [01-12|15:07:20.006] Loaded most recent local header
INFO [01-12|15:07:20.006] Loaded most recent local full block
INFO [01-12|15:07:20.007] Loaded most recent local fast block
INFO [01-12|15:07:20.007] Loaded local transaction journal
INFO [01-12|15:07:20.007] Regenerated local transaction journal
INFO [01-12|15:07:20.028] Allocated fast sync bloom
INFO [01-12|15:07:20.137] New local node record
INFO [01-12|15:07:20.139] IPC endpoint opened
INFO [01-12|15:07:20.164] Started P2P networking
e6e2aa8b4c27fef1e318aac755795325091e18563bbaf660ab9f90d3068178b87c@127
WARN [01-12|15:07:20.369] Served eth_coinbase
Welcome to the Geth JavaScript console!

```

Figure 3.58 Geth Console

It will show sync messages from time to time. Since it's being synced with some Blockchain. To avoid that, you can use the following command. 'Geth --verbosity 0 console'. Changing the level of verbosity adjusts the sync logs being displayed.

```

naveed@naveed-VirtualBox:~$ geth --verbosity 0 console
Welcome to the Geth JavaScript console!

instance: Geth/v1.9.9-stable-01744997/linux-amd64/go1.13.4
at block: 0 (Wed, 31 Dec 1969 17:00:00 MST)
datadir: /home/naveed/.ethereum
modules: admin:1.0 debug:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0
>

```

Figure 3.59 Geth Console with Verbosity

For geth client to work, we have to make it a node in decentralized network. Node.js is open source platform where we can execute java script code for practice purposes. NPM is the Node.js package system. It is the largest ecosystem of open source libraries in world. I am installing nvm. NVM stands for Node.js version manager.

First we will run this command, 'sudo apt-get update' It is used for updating local system and getting to know what updates are available from internet.

Second command 'sudo apt-get install build-essential libssl-dev' This to get all the required components to compile the code in any other language. It also contains some libraries and utilities.

```
> naveed@naveed-VirtualBox:~$ sudo apt-get update
[sudo] password for naveed:
Hit:1 http://ca.archive.ubuntu.com/ubuntu bionic InRelease
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:3 http://ca.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Hit:4 http://ppa.launchpad.net/ethereum/ethereum/ubuntu bionic InRelease
Get:5 http://ca.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:6 http://ca.archive.ubuntu.com/ubuntu bionic-updates/main amd64 DEP-11 Metadata [295 kB]
Get:7 http://ca.archive.ubuntu.com/ubuntu bionic-updates/main DEP-11 48x48 Icons [73.8 kB]
Get:8 http://ca.archive.ubuntu.com/ubuntu bionic-updates/main DEP-11 64x64 Icons [143 kB]
Get:9 http://ca.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 DEP-11 Metadata [264 kB]
Get:10 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metadata [38.5 kB]
Get:11 http://security.ubuntu.com/ubuntu bionic-security/main DEP-11 48x48 Icons [17.6 kB]
Get:12 http://security.ubuntu.com/ubuntu bionic-security/main DEP-11 64x64 Icons [41.5 kB]
Get:13 http://ca.archive.ubuntu.com/ubuntu bionic-updates/universe DEP-11 48x48 Icons [196 kB]
Get:14 http://security.ubuntu.com/ubuntu bionic-security/universe amd64 DEP-11 Metadata [42.1 kB]
Get:15 http://ca.archive.ubuntu.com/ubuntu bionic-updates/universe DEP-11 64x64 Icons [438 kB]
Get:16 http://security.ubuntu.com/ubuntu bionic-security/universe DEP-11 64x64 Icons [109 kB]
Get:17 http://security.ubuntu.com/ubuntu bionic-security/multiverse amd64 DEP-11 Metadata [2,464 B]
Get:18 http://ca.archive.ubuntu.com/ubuntu bionic-updates/multiverse amd64 DEP-11 Metadata [2,464 B]
Get:19 http://ca.archive.ubuntu.com/ubuntu bionic-backports/universe amd64 DEP-11 Metadata [7,980 B]
Fetched 1,924 kB in 6s (325 kB/s)
Reading package lists... Done
naveed@naveed-VirtualBox:~$ sudo apt-get install build-essential libssl-dev
Reading package lists... Done
Building dependency tree
```

Figure 3.60 Installing Libraries required for Geth

'Sudo apt install curl' is used to install curl program in Ubuntu. Curl is a tool to transfer or download files and data from any other machine using HTTP, FTP, HTTPS, Telnet, TFTP, IMAP. In this lab, it is used to fetch the file from another website and install it on the local machine. You can also use the curl to check whether the given URL is valid or not. It can be used to download the webpage as well.

```
naveed@naveed-VirtualBox:~$ sudo apt install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcurl4
The following NEW packages will be installed:
  curl libcurl4
0 upgraded, 2 newly installed, 0 to remove and 245 not upgraded
```

Figure 3.61 Installing Curl

Following commands are all documented on this URL for installing node.js.

<https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-18-04#installing-using-nvm> Curl is used to download nvm installation scripts from GitHub website.

‘vi install_nvm.sh’ Nano editor is used to view the scripts downloaded for installation. They should be alright. Nano is a built in text editor for UNIX based systems. It includes all the functionalities like spellchecking, syntax check and encoding.

‘bash install_nvm.sh’ This command will install the software into a subdirectory of your home directory. It will also add the necessary lines to your file. Bash is used to run the script. Bash is the most compatible interpreter that can run the commands from any file. Bash is built-in feature of Ubuntu in newer versions.

You can either log out or log back in to gain access of nvm functionality or use the Next command ‘source ~/.profile’ so that the current session will know about the change.

Now that nvm is installed. We can install isolated node.js versions from this command

‘nvm ls-remote’. You will get a list of all the versions available. Install the latest LTS version.

```

naveed@naveed-VirtualBox:~$ curl -sL https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.sh -o install_nvm.sh
naveed@naveed-VirtualBox:~$ vi install_nvm.sh

/bin/bash: q: command not found

shell returned 127

naveed@naveed-VirtualBox:~$ bash install_nvm.sh
=> Downloading nvm as script to '/home/naveed/.nvm'

=> Appending nvm source string to /home/naveed/.bashrc
=> Appending bash_completion source string to /home/naveed/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
naveed@naveed-VirtualBox:~$
naveed@naveed-VirtualBox:~$ source ~/.profile
naveed@naveed-VirtualBox:~$ nvm ls-remote
v0.1.14
v0.1.15

```

Figure 3.62 Install node.js - 1

Use the ‘nvm install 8.17.0’ command to install.

‘node -v’ lets us know if the node is installed and what version is being used.


```
naveed@naveed-VirtualBox:~$ nvm install 8.17.0
Downloading and installing node v8.17.0...
Downloading https://nodejs.org/dist/v8.17.0/node-v8.17.0-linux-x64.tar.xz...
#####
Computing checksum with sha256sum
Checksums matched!
Now using node v8.17.0 (npm v6.13.4)
Creating default alias: default -> 8.17.0 (-> v8.17.0)
naveed@naveed-VirtualBox:~$ node -v
v8.17.0
```

Figure 3.63 Install node.js - 2

If you want to run smart contracts on geth, we will need to install compiler for solidity language. Solidity is the most common language to be used for smart contracts. It gives output in bytecode which can be interpreted by Ethereum Virtual Machine. Solidity code is written in human readable files with extension sol. Solc is compiler for solidity. It is built by using C++. Solc is in the default package manager of node.js. As we have already installed npm, we can run this command ‘npm install –g solc’. [42]

```
naveed@naveed-VirtualBox:~$ npm install -g solc
/home/naveed/.npm/versions/node/v8.17.0/bin/solcjs -> /home/naveed/.npm/versions
/node/v8.17.0/lib/node_modules/solc/solcjs
+ solc@0.6.1
added 25 packages from 15 contributors in 4.475s
naveed@naveed-VirtualBox:~$
```

Figure 3.64 Installing Compiler for Solidity

This is a very simple smart contract written in solidity. It is written in normal text editor on Ubuntu system and saved on the desktop with extension sol.

1. Pragma is generally the first line of any solidity code. This command specifies which version of compiler is being used. In this particular code, version is 0.6.1.
2. Second line defines a contract, which is similar to class in other languages. A contract consists of multiple constructs, that are defined under contract.

3. Third line is state variable. Variables declared in a contract that are not used in function are called state variables. State variable is used for storing the current value of contract. State variable defined as mapping between integer and address. Address is a 20-byte value of an Ethereum user account or contract account. Mapping is similar to hash tables.
4. Fourth line is contract constructor. Constructor is only executed once while deploying contract.
5. Fifth lines show that 10,000 SimpleCoin tokens are being assigned to coin account with that address.
6. Functions are used to read/write values in state variables. Functions are executed on demand in code. Sixth line says to move the number of simplecoin tokens from the coin account of the function caller to the specified coin account.
7. Decrement in the coin balance of sender account.
8. Increment in the coin balance of receiver account.

Contract in solidity is similar to class in any other language. coinBalance is a state variable. It is defined as mapping. Mapping is a hash function where key here is address whose value is unsigned 256-bit integer. Address can hold up to 20-byte value. So coinBalance represents coin account containing tokens. Transfer function is used to transfer coins from one account to other specified account. Transfer function says to subtract the amount of coins from the function caller account and add the coins into account specified. Msg.sender is the address of account sending the coins. Below table shows the keywords being used in this code. [47]

Keyword	Explanation
contract	Type similar to class in any other language
mapping	Data structure similar to a hash table or hash map
address	20-byte value representing an Ethereum user account or contract account
uint256	Unsigned 256-bit integer
msg	Special variable representing an incoming message object
msg.sender	Property of the msg object representing the address of the message sender

Table 6 Keywords in Code

[48]

It's a simple code to transfer simple coin from one account to another. Earlier we have used this in remix compiler and see the output of this in GUI.

```

pragma solidity ^0.6.1;
contract SimpleCoin {
    mapping (address => uint256) public coinBalance;
    constructor() public {
        coinBalance[0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C] = 10000;
    }
    function transfer(address _to, uint256 _amount) public {
        coinBalance[msg.sender] -= _amount;
        coinBalance[_to] += _amount;
    }
}

```

Figure 3.65 Code Written

[43]

Now we have to compile this code into ABI (application binary interface) and BIN (bytecode) for deploying into Blockchain. Following commands are used to compile the code in bin and abi. You have to be specified location to compile the code. Since my sol file was on desktop so it was compiled and stored the outputs in desktop.

```
naveed@naveed-VirtualBox: ~/Desktop$  
naveed@naveed-VirtualBox: ~/Desktop$ solcjs --abi simplecoin.sol  
naveed@naveed-VirtualBox: ~/Desktop$ solcjs --bin simplecoin.sol  
naveed@naveed-VirtualBox: ~/Desktop$  
naveed@naveed-VirtualBox: ~/Desktop$
```

Figure 3.66 Compiling Code in bin and abi files

If there are no errors in code, it will get compiled and will be stored in abi and bin files on desktop.



Figure 3.67 Result after compiling

Now you have to run the following commands to see the outputs of both files.

```
more simplecoin_sol_SimpleCoin.abi  
more simplecoin_sol_SimpleCoin.bin
```

Solidity is a case sensitive language. You have to carefully see the results.

This code assigns ten thousand ether tokens to the account whose address is mentioned. Then it transfers those tokens to another account causing decrement in first account and increment in second account. We can clearly observe the outputs from remix compiler earlier in this chapter.

```
naveed@naveed-VirtualBox:~/Desktop$ more simplecoin_sol_SimpleCoin.abi
[{"inputs": [], "stateMutability": "nonpayable", "type": "constructor"}, {"inputs": [{"internalType": "address", "name": "", "type": "address"}], "name": "coinBalance", "outputs": [{"internalType": "uint256", "name": "", "type": "uint256"}], "stateMutability": "view", "type": "function"}, {"inputs": [{"internalType": "address", "name": "_to", "type": "address"}, {"internalType": "uint256", "name": "_amount", "type": "uint256"}], "name": "transfer", "outputs": [], "stateMutability": "nonpayable", "type": "function"}]
naveed@naveed-VirtualBox:~/Desktop$ more simplecoin_sol_SimpleCoin.bin
60806040523480156100115760006000fd5b505b612710600060005060007314723a09acff6d2a60
dcd77aa4aff308fddc160c73ffffffffffffffffffffffffffffffffffffffff1673ffffffffffff
ffffffffffffffffffffffff1681526020019081526020016000206000508190909055505b61
007b565b6101e78061008a6000396000f3fe60806040523480156100115760006000fd5b50600436
1061003b5760003560e01c8063a9059cbb14610041578063fabde80c146100905761003b565b6000
6000fd5b61008e600480360360408110156100585760006000fd5b81019080803573ffffffffffff
ffffffffffffffff169060200190929190803590602001909291905050506100e956
5b005b6100d3600480360360208110156100a75760006000fd5b81019080803573ffffffffffff
ffffffffffffffff169060200190929190505050610196565b60405180828152602001
91505060405180910390f35b80600060005060003373fffffffffffffffffffffffffffff
ffff1673ffffffffffffffff16815260200190815260200160002060
0082828250540392505081909090555080600060005060008473fffffffffffffffffffff
ffffffff1673ffffffffffffffff168152602001908152602001
60002060008282825054019250508190909055505b5050565b600060005060205280600052604060
0020600091509050548156fea2646970667358221220d523d52b733a1d4e653d8d8bf1081ce23f59
2289133a68db1fa89b848e9956fa64736f6c63430006010033
naveed@naveed-VirtualBox:~/Desktop$
```

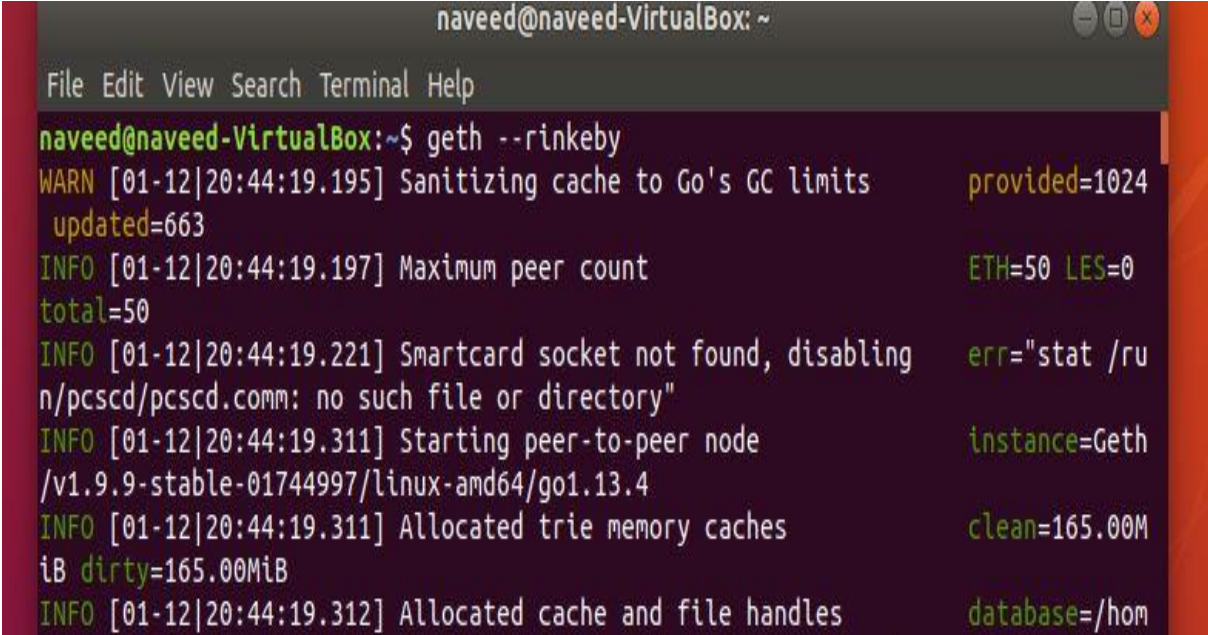
Figure 3.68 Displaying contents of result file

You can synchronize your geth node with any test networks. Most famous are ropsten, kovan and rinkeby. I will be synchronizing my geth node with rinkeby network. Rinkeby is also supported on etherscan. Etherscan will be discussed in detail later in this chapter. The link below shows displays the gasprice, gaslimit, when was last block mined and instructions on how to connect local geth node to rinkeby using any operating system. Blocks are generated at an interval of 15 seconds in rinkeby. The consensus algorithm used in rinkeby is proof of algorithm. POA proof of algorithm is same in many ways to proof of work POW, but it has more control over the miners. This means the miner have to identify himself in order to receive ether.

<https://www.rinkeby.io/#stats>

<https://www.rinkeby.io/#faucet>. This is the ether faucet linked with rinkeby. This is used to prevent malicious actors from using all the available funds or gathering enough ether to encourage spam attacks. All the requests are tied to social network accounts and have a permitted limit to gather ether. Rinkeby has testing ether coins. They don't have any value in real world. It is not allowed to send ether from main net to the test net. This allows the developers to test contracts on Blockchain without having the fear of losing actual money.

Synching with test network can take up to two to three days depending upon internet speed and processing power of local machine. The last block synced with my node is 2 years and 6 months old. I have to stop it because it will download all the Blockchain. My local machine is not capable of handling this much because of less RAM and storage. You can stop the synching process by pressing control and D key simultaneously.



```
naveed@naveed-VirtualBox: ~
File Edit View Search Terminal Help
naveed@naveed-VirtualBox:~$ geth --rinkeby
WARN [01-12|20:44:19.195] Sanitizing cache to Go's GC limits      provided=1024
  updated=663
INFO [01-12|20:44:19.197] Maximum peer count      ETH=50 LES=0
total=50
INFO [01-12|20:44:19.221] Smartcard socket not found, disabling  err="stat /ru
n/pcscd/pcscd.comm: no such file or directory"
INFO [01-12|20:44:19.311] Starting peer-to-peer node      instance=Geth
/v1.9.9-stable-01744997/linux-amd64/go1.13.4
INFO [01-12|20:44:19.311] Allocated trie memory caches      clean=165.00M
iB dirty=165.00MiB
INFO [01-12|20:44:19.312] Allocated cache and file handles    database=/hom
```

Figure 3.69 Connecting to Rinkeby test Network - 1

```
eed/.ethereum/rinkeby/geth.ipc
INFO [01-12|21:15:55.364] Imported new block headers          count=384  el
apsed=224.191ms number=602201 hash=6fc564...64e9c9 age=2y6mo23h
INFO [01-12|21:15:55.423] Imported new state entries          count=2232 el
apsed=12.303ms  processed=1014718 pending=6070  retry=0  duplicate=0 unexpected
=0
^C WARN [01-12|21:15:55.521] Already shutting down, interrupt more to panic. time
s=9
INFO [01-12|21:15:55.531] Imported new block receipts        count=186  el
apsed=286.835ms number=601817 hash=9816d6...595611 age=2y6mo1d size=155.54KiB
INFO [01-12|21:15:55.535] Blockchain manager stopped
INFO [01-12|21:15:55.542] Stopping Ethereum protocol
INFO [01-12|21:15:55.561] Ethereum protocol stopped
INFO [01-12|21:15:55.561] Transaction pool stopped
CRIT [01-12|21:15:55.599] Failed to store last fast block's hash  err="leveldb:
closed"
naveed@naveed-VirtualBox:~$
```

Figure 3.70 Connecting to Rinkeby Test network - 2

You can create new accounts on geth node. With new account creation, you have to assign a password to it. You will be required to enter a password twice. This task is being done on console, it can be done on geth command line as well. Commands are different in both types. Two different accounts are created and they are listed below by 'eth.accounts' command. Below screenshots show the results of this command.

You can check the balance in accounts whether in Wei or in Ether. Transferring ether from one account to another is also possible. You will be asked to enter the password for transfer to complete.

You can start mining ether by this command 'miner.start().' It can be stopped by 'miner.stop()'. If the value returned for the command of start mining is null, this means your node is not successfully synced with test net.

```
[ ]
> personal.newAccount()
Password:
Repeat password:
"0x55d4a6828e6da97c82a8e4a56f4d2cf8d83ab1aa"
>
```

Figure 3.71 Creating new accounts in Geth

```
[ 0x55d4a6828e6da97c82a8e4a56f4d2cf8d83ab1aa ]
P> personal.newAccount()
P Password:
C Repeat password:
E "0xa9e1883adb200c5a98f8c33673a784db034fc060"
C
P> eth.accounts
E ["0x55d4a6828e6da97c82a8e4a56f4d2cf8d83ab1aa", "0xa9e1883adb200c5a98f8c33673a784db034fc060"]
E
P>
```

Figure 3.72 Displaying Created accounts in Geth

3.7 Issues with Smart Contracts: -

A fully working smart contract should have the following properties. Firstly, it should be executed automatically. Secondly, a Smart contract should enforce the terms and conditions set to its parties. It should be semantically correct. Smart contract should be secure and unstoppable because of the immutable property of Blockchain. [12]

Many of the developers around the world have reported the same kind of issues they face while developing smart contracts. These issues are classified into four categories, coding, security, privacy, and performance issues.

Smart contract cannot be modified or terminated once it is deployed due to the stability feature of Blockchain. But as per real law in the world, contracts should be modified as per needs. So there should be a set of standards that need to be followed to modify or terminate smart contracts. Smart contracts are written in procedural languages. Procedural language follows an order while executing a set of commands; examples are C, Fortran. So developer before developing smart contract has to specify the steps and order of code, which would dispose of it to errors. Critics say that to avoid this problem smart contracts should be based on logic-based languages rather than procedural.

Performance issue is related to the number of smart contracts executing at a given time in Blockchain. Smart contracts have to be executed in sequential order in Blockchain so one smart contract can be executed at a given time. This causes delays and performance issues concerning the rising popularity of contracts.

The main privacy issue of smart contracts is the lack of transactional security. Anyone anywhere in the world can see the transactions and modify it without any approval. Smart contracts are mainly used in financial transactions and the confidentiality of those

transactions is the main concern of the owner. Critics claim that the use of restriction codes can prevent any unknown user to modify it. Restriction code will lead to do the transaction in private and only the concerned parties can control it. Second issue with privacy is data feeds privacy. Whenever a contract needs data from Blockchain, it's requesting is broadcasted to all over Blockchain. This means that any unknown user can make use of this information. [13]

There could be many security vulnerabilities of a smart contract. External call, untrustworthy data feeds, Gasless sends, Ether lost, Immutable bugs, Block Hash usage, timestamp dependency and re-entrancy. Externals call means calling to external contracts or unknown contracts can introduce malicious code in contract to be executed.

Sometimes contracts require data from systems outside the Blockchain. These systems could be website that guarantee no trustiness of the information provided. This is the problem with untrustworthy data feeds. Every transaction in Ethereum requires a fee. So if you are trying to do a transaction with insufficient gas in account, transaction will not be done but your gas will be all spent. Ether loss occurs when ether is sent to a recipient that has no association with any contract or user. Ether sent to such a contract can result in loss. There would be no recovery of such ether. Re-entrancy was exploited in The DAO attack, which will be explained later on. Immutable bugs refer to the property of immutability in Blockchain. If the contract deployed has a bug in it, this means no one can change it and it will get executed that can cause loss of ether. Timestamp dependency refers to a miner who will call a function in contract to execute a transaction. But a dishonest miner can manipulate that timestamp in his favour. Therefore, it is recommended that no dependency of the timestamp can be considered after fifteen minutes. Block hash can be used by miners just like timestamps to refrain important components of a block to execute. [14]

Despite all of these issues, smart contracts are gaining popularity in both public and private domains. The main feature of Blockchain that lets parties deal with each other without a middleman is appealing to the world. Due to peer to peer nature of Blockchain, letting the ordinary user control their and exchange rate of cryptocurrency has given a new face to modern technology. Smart contract deployed on top of this Blockchain makes it more feasible. Efforts are underway to make smart contracts use in elections, supply chain, IoT and insurance other than just banking.

3.8 The DAO Attack: -

With the invention of smart contract, a term named dynamic autonomous organization DAO also came into being. DAO is made by combining multiple smart contracts by running a certain code and then that code functions on a governance mechanism. DAO main purpose is to digitalize the rules and regulations of certain organization and removing the need for documentation and governance manually. [67] [68] A basic DAO works as follows:

Number of people in an organization writes the smart contracts. These smart contracts will be termed by DAO. Funding will be done by people in organization by purchasing tokens that represents ownership and to provide the resources to DAO. As soon as funding is finished, DAO starts to operate. Then those people can make proposals to DAO on how to spend money and the members who funded it will approve those proposals. [69]

DAO was launched on 30th April 2016 having a 28-day funding period. DAO raised more than 150 million in the largest crowdfunding ever in history. Several members of the funding expressed their concern that the code was vulnerable to attackers. It was assured that funds were not at risk admitting that there is a recursive call bug that is going to be fixed soon. Unfortunately, while it was getting fixed, an unknown hacker exploited this bug and stole more than 50 million dollars by making a child DAO of the original DAO. Child DAO created by an attacker has same regulations, structure and limitations as parent DAO and no one can get the ether in child DAO in 28 days because that was the funding period. That was the large time window for the attacker if he wanted to cash out those ethers. [70]

The attacker made use of fallback function in DAO attack. He found a loophole is a split procedure. Split has the ability to part the DAO into two and the voters who voted to do this split will move their ether from parent DAO to this newly created child DAO. The split procedure can be called by any token holder. But once anyone decides to do split procedure, he needs at least 48 days to move the ether from child DAO to any account user control. Loophole was that when the split function is called, attacker can retrieve the ether first and update the balance later. Moreover, it wasn't checking whether it can be recursive call, that the function calls itself. So the attacker managed to call a recursive

function and receive the funds multiple times before getting to the step to check the balance. [71]

DAO attack works in this way simplified: - [72]

- Call a split as a recursive function and wait till the voting period ends.
- Perform split regularly.
- Parent DAO will send the tokens to child DAO.
- Work in such a way that parent DAO don't have enough time to update the record and it would send tokens anyway.
- Parent DAO will send more and more tokens.
- Withdraw your reward before it actually updates the balance.
- Again let the parents DAO run the split function. In this way it would never update the balance.

DAO is the first big scale application of Ethereum. There is a lot of risks involved in programming, smart contracts as they require transactions of real money and they can be exploited by hackers by finding loopholes. Ethereum is still in its developing phase and new updates are being added to this Blockchain on regular basis.

3.9 Future of blockchain based Smart Contracts: -

The future of smart contracts is very encouraging, keeping in view today's culture. A lot of banks have already implemented smart contracts to improve processing times and are clearing payments through automation made possible by smart contracts. There are many cryptocurrencies that handle smart contracts. Most notable is Ethereum, along with bitcoin, rootstock, and ripple. Smart contracts provide properties like autonomy, trust, backup, safety, speed, savings, and accuracy.

Smart contracts will have a strong impact on certain industries like law and financial institutions. Smart contracts are suggested to replace lawyers in near future. Because there is no need for a judge to settle a contract dispute. These are committed codes in blockchain which no one can dispute. Lawyers will transfer from writing traditional contracts to standardized smart contracts that will remain on the blockchain forever and both parties will have to follow that no matter the circumstances. A supermarket named 'carrefour' has implemented smart contracts to guarantee the quality of milk. Brands like

Louis Vutton have also integrated blockchain into their systems to keep a check on their products and to eliminate counterfeit products. Smart contracts have the property of immutability which means that no information can be changed or deleted once the contract is activated. This property is particularly useful for the law industry as most of the rules remain the same over the years and smart contracts will keep the integrity in place.

The insurance industry might be affected by smart contracts the most in the future. In present times, most of the people don't bother to buy insurance as they don't have sufficient trust in system. Smart contracts can reshape the interaction between insurance buyers and sellers by replacing human need in contracts with automated decentralized infrastructure. Blockchain can keep multiple reliable records of claims that can prevent fraudulent parties from filling multiple false claims for same incident. Smart contract can automatically trigger pay-outs based on external data. [79]

Smart Contracts are being effectively used in the utilities industry. A microgrid can be maintained by smart contract. Blockchain technology will significantly reduce inaccuracies in distribution network due to the verification system which requires real time data matching between peer to peer. Smart contract maintains the physical health of microgrid by distributing power on real time usage. [80]

Blockchains are applied in different streams because of its ability to automate. Smart contracts are the main part of this automation process. Smart contracts leverage code to automatically execute between two parties, eliminating the need for human intervention in the process. For example, if a car transaction system runs on a blockchain, there must be certain credit checks or previous driving history conditions to be met before handing over the car to customer. Smart contract makes sure all the conditions are fulfilled otherwise it will trigger an alarm notifying that transaction can't be made. [81]

There are certain risks associated with smart contracts like security and costs related with those transactions. If these risks could be controlled or reduced in future, smart contracts will take over many fields of life for sure.

Smart contracts are being used in field of artificial intelligence. Cortex is decentralized artificial intelligence platform that supports smart contracts execution. AI developers can upload their models on blockchain and dApp developer can then access these models by certain crypto currency. Integrating smart contracts with AI can be a major breakthrough

for smart contracts. Cortex is the first infrastructural blockchain to support AI. It will help the developers in making open source community for making more AI smart contracts. The entire ecosystem will have a platform supporting AI and smart contracts. With this unique approach of cortex, there is no need for third party interface. It is combining AI interface directly on blockchain. Blockchain used here is Ethereum and solidity is the programming language for writing codes. [82]

There is still a long way to go before everything is governed by smart contracts. Smart contracts do provide a certain type of relief to traditional systems hanging around but factors like cost and breach of security hasn't been fully addressed. Creative solutions should be kept in place to deal with risks that arise with smart contracts deployment in real life. Main examples of industries that can be transformed with smart contracts are identity management, banking/payments, supply chain, real estate and healthcare records.

4. Conclusion

The last section of my report is about the conclusion I have drawn from my research on Blockchain and specifically smart contracts.

In the first chapter, I have discussed the history of Blockchain and the invention of the first ever Blockchain called bitcoin. One successful consensus algorithm led to the creation of first-ever cryptocurrency world has ever known called bitcoin. Different types of blockchains are discussed. Proof of work consensus algorithm was used in bitcoin, but with Ethereum, a new consensus algorithm called proof of stake was developed. POW has certain vulnerabilities when it comes to the security of Blockchain. Due to which researchers developed a new, more protected consensus algorithm. There are many blockchains that support the use of smart contracts, but Ethereum is the most popular one and widely used for deploying smart contracts. That's why I have used Ethereum in my capstone. With the use of smart contracts, decentralized applications are developed which specifically work on Blockchain.

Public and private blockchains are discussed in detail. The advantages, disadvantages and algorithms used in these blockchains and how they are significantly different from each other are mentioned. Blockchain eliminated the use of double spending in our normal life transactions. Future of Blockchain along with its use in the financial industry, content distribution networks, social networks, domain name system, decentralized internet and cybersecurity, are discussed in chapter 1. Different platforms are available other than Ethereum where you can work with smart contracts. The most famous ones discussed in the report are RSK, Cardano and EOS.

Second chapter is more specific on technical details about Ethereum and ether as a cryptocurrency works. Different components of Ethereum Blockchain like ether, gas, accounts and EVM are mentioned in detail. POS algorithm used in Ethereum consumes less energy as compared to POW in bitcoin. POS provides a new method of adding new block to chain that is more secure and reliable. Rapid mining led to centralization of bitcoin and it was mainly connected with POW. Concept of gas is introduced in Ethereum which is very different from other blockchains. Gas is used to regulate smart contracts being deployed so that they don't run in infinite loop and consume all the resources. Every smart contract has specific gas associated with it. EVM is the main

component in Ethereum Blockchain. EVM works as an operating system for Ethereum Blockchain. Different types of accounts that exist in Ethereum Blockchain are discussed in this chapter.

Third chapter of my capstone is specifically focused on smart contracts. Programming language used for smart contracts in Ethereum is solidity. Solidity is a Turing complete language. Different parts of the solidity code are explained in detail. This chapter comprises of lab component of my capstone. What are the efficient and easy ways to deploy smart contracts on Ethereum Blockchain, how can we run a test network from host computer or connect to real Blockchain are shown with commands and screenshots and deployed smart contracts interaction with Blockchain and how does it makes a difference that regular code is explained. User friendly remix compiler is used to compile the contract into ABI and bytecode. Code is written in solidity language. Go Ethereum is installed in Linux machine and it is connected to the test network from Linux machine.

Issues with smart contracts and DAO is explained in the last chapter. Issues mentioned are of significant importance as they are the reason smart contracts are not being used widely. The future of smart contracts is explained. MetaMask is the browser application that lets you interact with Blockchain from your host machine. Installation of MetaMask and the use of fake mined ether is shown in this chapter.

Smart contracts in Ethereum are still at very early stages. Ether is not still as popular as bitcoin. They are still a lot of improvements to be made before using them widely in general public. Privacy, security and immutability are the main issues related to smart contracts now. If these issues are tackled, smart contracts will be very popular in near future. Smart contracts in Ethereum are easy to code and deploy. All the transactions done in Ethereum Blockchain is visible to everyone around the world. Since the operation of the smart contract shows trust between different parties without the need for an intermediary is a great sign for normal users. Many fields like medical, law and banking can be revolutionized in the near future by the use of smart contracts. Blockchain in particular are gaining popularity because of the massive success of cryptocurrencies. Some researchers believe that smart contracts and cryptocurrencies will replace traditional contracts and traditional currencies in the future.

LIST OF TABLES: -

Table 1	Depiction of how PBFT works
Table 2	Different Wallet Types
Table 3	Instructions and their gas costs
Table 4	Opcodes in EVM
Table 5	Common Function Types in solidity
Table 6	Keywords in Code
Table 7	Differences between Smart contract Platforms

LIST OF FIGURES: -

Fig. 1.1	Proof of work Architecture
Fig. 1.2	Comparison of Centralized and Decentralized System
Fig. 1.3	Huawei's Blockchain Hierarchy
Fig. 1.4	Representation of Two Nodes
Fig. 1.5	Demonstration of Nodes Agreeing to Add Another Node
Fig. 1.6	Demonstration of How Smart Contract Works
Fig. 1.7	Blockchain Layers and Blockbench Workload
Fig. 1.8	Pie Chart of Blockchain Applications
Fig. 2.1	Depiction of POS Works - 1
Fig. 2.2	Depiction of POS Works - 2
Fig. 2.3	Depiction of POS Works - 3
Fig. 2.4	Ether Life Cycle
Fig. 2.5	EVM Block Structure
Fig. 2.6	Block Diagram of V-ROM and Stack Of EVM
Fig. 2.7	Execution Environment in Ethereum Blockchain
Fig. 2.8	Types of Accounts
Fig. 3.1	Block Representation Of How Compiler Works
Fig. 3.2	Sample Code - 1
Fig. 3.3	Sample Code - 2

Fig. 3.4	MetaMask Setup in FireFox – 1
Fig. 3.5	MetaMask Setup in FireFox – 2
Fig. 3.6	MetaMask Setup in FireFox – 3
Fig. 3.7	Password Creation in MetaMask
Fig. 3.8	Backup Protection in MetaMask
Fig. 3.9	Main Window of MetaMask
Fig. 3.10	Main net and Test nodes
Fig. 3.11	Display of how Rinkeby Faucet Works – 1
Fig. 3.12	Display of how Rinkeby Faucet Works – 2
Fig. 3.13	Twitter Window
Fig. 3.14	Rinkeby Faucet
Fig. 3.15	Successful transfer
Fig. 3.16	Block Diagram of How Remix Works
Fig. 3.17	Remix Front Page
Fig. 3.18	List of Common plugins in Remix
Fig. 3.19	Activate plugins
Fig. 3.20	Old Remix Front page
Fig. 3.21	Remix Front page with written code
Fig. 3.22	Right side tabs of remix
Fig. 3.23	Code written in Remix
Fig. 3.24	Compiling Options

Fig. 3.25	Errors after running the code
Fig. 3.26	Contract deployed on Blockchain
Fig. 3.27	Output of contract
Fig. 3.28	Console Messages
Fig. 3.29	Experimenting with code – 1
Fig. 3.30	Experimenting with code – 2
Fig. 3.31	Experimenting with code – 3
Fig. 3.32	Experimenting with code – 4
Fig. 3.33	Experimenting with code – 5
Fig. 3.34	Experimenting with code – 6
Fig. 3.35	Experimenting with code – 7
Fig. 3.36	Experimenting with code – 8
Fig. 3.37	Second Code
Fig. 3.38	MetaMask Display Message - 1
Fig. 3.38	Output of Code
Fig. 3.40	MetaMask Display Message – 2
Fig. 3.41	Tabs after compiling code
Fig. 3.42	MetaMask Display Message – 3
Fig. 3.43	Etherscan – 1
Fig. 3.44	Etherscan – 2
Fig. 3.45	MetaMask Display Message – 4

Fig. 3.46	MetaMask Display Message – 5
Fig. 3.47	MetaMask Display Message – 6
Fig. 3.48	MetaMask Display Message – 7
Fig. 3.49	MetaMask Display Message – 8
Fig. 3.50	MetaMask Display Message – 9
Fig. 3.51	Code Written
Fig. 3.52	Function Explained
Fig. 3.53	Working of our Contract
Fig. 3.54	Geth Communication types
Fig. 3.55	Installing Geth on Ubuntu - 1
Fig. 3.56	Installing Geth on Ubuntu - 2
Fig. 3.57	Connecting to Main Blockchain
Fig. 3.58	Geth Console
Fig. 3.59	Geth console with verbosity
Fig. 3.60	Installing libraries required of Geth
Fig. 3.61	Installing Curl
Fig. 3.62	Installing node.js – 1
Fig. 3.63	Installing node.js – 2
Fig. 3.64	Installing compiler for solidity
Fig. 3.65	Code Written
Fig. 3.66	Compiling code in bin and abi files

Fig. 3.67	Result after compiling
Fig. 3.68	Displaying contents of result files
Fig. 3.69	Connecting to Rinkeby test network – 1
Fig. 3.70	Connecting to Rinkeby test network – 2
Fig. 3.71	Creating new accounts in Geth
Fig. 3.72	Displaying Created accounts in Geth

5. Works Cited

- [4,5,12,50,51,52,56,59,60,63,64] Bashir, I. (n.d.). *Mastering Blockchain*.
- [10] Beyer, S. (2019). <https://www.mycryptopedia.com/ethereum-whisper-a-detailed-guide/>. Retrieved from MycryptoPedia.
- [6,7] Bruno Sabio de albuquerque, M. C. (n.d.). *Understanding Bitcoins: Facts and Questions*.
- [14] Flatscher, A. M. (2018). *Security Vulnerabilities in Ethereum Smart Contracts*.
- [11,13] Hassan, M. A. (2019). *Blockchain and Smart contracts*.
- [8] Kulkarni, K. (n.d.). *Learn Bitcoin and Blockchain*.
- [1,2,3] Nakamoto, S. (2008). *A peer to peer Electronic Cash System*.
- [9,15,16,40] Steele, J. (2018). *Blockchain applications and Smart Contracts: Developing With Ethereum and solidity*.
- [17,38,43] Modi, R. (2018). *Solidity Programming Essentials*
- [18] Consensus Algorithm for private Blockchain
- [19] <https://www.mangoresearch.co/casper-nothing-at-stake-problem/> Shawn Dexter. 2018
- [20] <https://www.forbes.com/sites/kpmg/2018/09/11/blockchain-and-the-future-of-finance/#54b37cdb620f/> John E. Mulhall
- [21,22,23,24,25,26] Blockchain and the future of internet: A comprehensive review, Feb 2019, by Fakhur ul Hassan, Anwaar Ali, Junaid Qadir
- [27,35,36,37,39,43,47,48] Roberto Inferno, *Building Ethereum Dapps*. March 2019
- [28] State of Public and Private Blockchains: Myths and Reality by C. Mohan, June 30 2019
- [29] BlockBench: Framework for analysing private blockchains, Ji Wang, Rui Li, Wang Chen March 2017

- [30] Huawei Blockchain Whitepaper
- [31] <https://crushcrypto.com/what-is-practical-byzantine-fault-tolerance/> Victor, October 2018.
- [32] Practical Byzantine Fault Tolerance, Michael Castro and Barbara Liskov, February 1999
- [33,34] <https://www.investopedia.com/tech/what-ether-it-same-ethereum/> Nathan Rief/ June 25, 2019
- [40] <https://blockgeeks.com/guides/ethereum-gas/> Ameer Rosic
- [41] https://www.mobilefish.com/developer/blockchain/blockchain_quickguide_ethereum_related_tutorials.html
- [42] <https://www.devteam.space/blog/how-to-deploy-smart-contract-on-ethereum/>
- [43] <https://www.youtube.com/watch?v=4CsH5xTxhSA/>
- [44] <https://www.bitdegree.org/learn/solidity-online-compiler/>
- [45,46] Udemy Course by Stephen. Ethereum and Solidity: Complete Developer's Guide
- [47] Visual Emulation of Ethereum Virtual Machine by Robert Norvill, Radu State, Andrea Cullen, Beltra Borja Fiz
- [48] <https://www.quora.com/What-is-meant-by-Turing-complete-languages-and-which-programming-languages-comes-under-this>
- [49] <https://kauri.io/a-deep-dive-into-the-ethereum-virtual-machine-vm/b4a6d12332bd4ad58535ac2d59d95dff/a>
- [53] <https://medium.com/coinmonks/ethereum-account-212feb9c4154> by Hu kenneth
- [54] Ethereum: A secure decentralized Generalised Transaction Ledger Byzantine Version by Dr. Gavin Wood : Founder of Ethereum and Parity
- [55] https://solidity.readthedocs.io/en/develop/natspec-format.html#_github_etherium_documentation

[57,58,61,62] Smart contracts: security patterns in ethereum eco system and solidity by Maximilian Wohrer and Uwe Zdun , published in 2018 international workshop on blockchain oriented software engineering

[65,66] <https://medium.com/@jeancvllr/solidity-tutorial-all-about-enums-684adcc0b38e> by Jean Cvllr , Aug 1. 2019

[67] <https://medium.com/@ogueluturk/the-dao-hack-explained-unfortunate-take-off-of-smart-contracts-2bd8c8db3562>

[68,69,70] <https://www.coindesk.com/understanding-dao-hack-journalists>

[71] <https://medium.com/@MyPaoG/explaining-the-dao-exploit-for-beginners-in-solidity-80ee84f0d470>

[72] <https://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/>

[73,75,76,77] <https://blockgeeks.com/guides/smart-contract-platforms-comparison-rsk-vs-ethereum-vs-eos-vs-cardano/>

[74] Cardano Official Website, <https://www.cardano.org/en/what-is-cardano/>

[78] <https://bitcoinexchangeguide.com/top-5-blockchain-smart-contract-platforms-other-than-ethereum-eth/>

[79] <https://blog.chain.link/the-power-of-smart-contracts-what-they-are-and-how-they-can-revolutionize-the-future/>

[80,81] <https://isg-one.com/consulting/blockchain/articles/smart-contracts-the-future-of-contracting>

[82] <https://hackernoon.com/ai-smart-contracts-the-past-present-and-future-625d3416807b>