E-Voting Application
Using Ethereum Blockchain

SUMMER TRAINING REPORT

Submitted by

Submitted to:

Mr. Farzil Kidwai

Assistant Professor

CSE Dept.

Submitted By:

Praveen Singh Rathore

Enrollment No. 20314802717

C-5 (Sem 7)

Computer Science Engineering

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

ROHINI, NEW DELHI

Maharaja Agrasen Institute of Technology

To Whomsoever It May Concern

I, **Praveen Singh Rathore**, Enrollment No. **20314802717**, a student of **Bachelors of Technology (CSE), a class of 2017-21, Maharaja Agrasen Institute of Technology, Delhi** hereby declare that the Summer Training project report entitled **"E-Voting Application Using Ethereum Blockchain"** is an original work and the same has not been submitted to any other Institute for the award of any other degree.

Date: 23 June, 2020

Place: Delhi

Praveen Singh Rathore

Enrollment No: 20314802717

Computer Science Engineering

C-456/C-5

ACKNOWLEDGEMENT

First and foremost, I wish to express my profound gratitude to Coursera for giving me the opportunity to carry out my training through such an informative course. I find great pleasure to express my unfeigned thanks to our Instructor Mrs. Beena Ramamurthy, Teaching Professor, Computer Science and Engineering, University at Buffalo & The State University of New York for her invaluable guidance, support and useful suggestions at every stage of this training course.

My heartfelt thanks to Mrs. Beena Ramamurthy for her immense help and support, useful discussions and valuable recommendations throughout the course of my training .

I wish to thank my peers and coursemates for their continuous discussions on various doubts throughout the course.

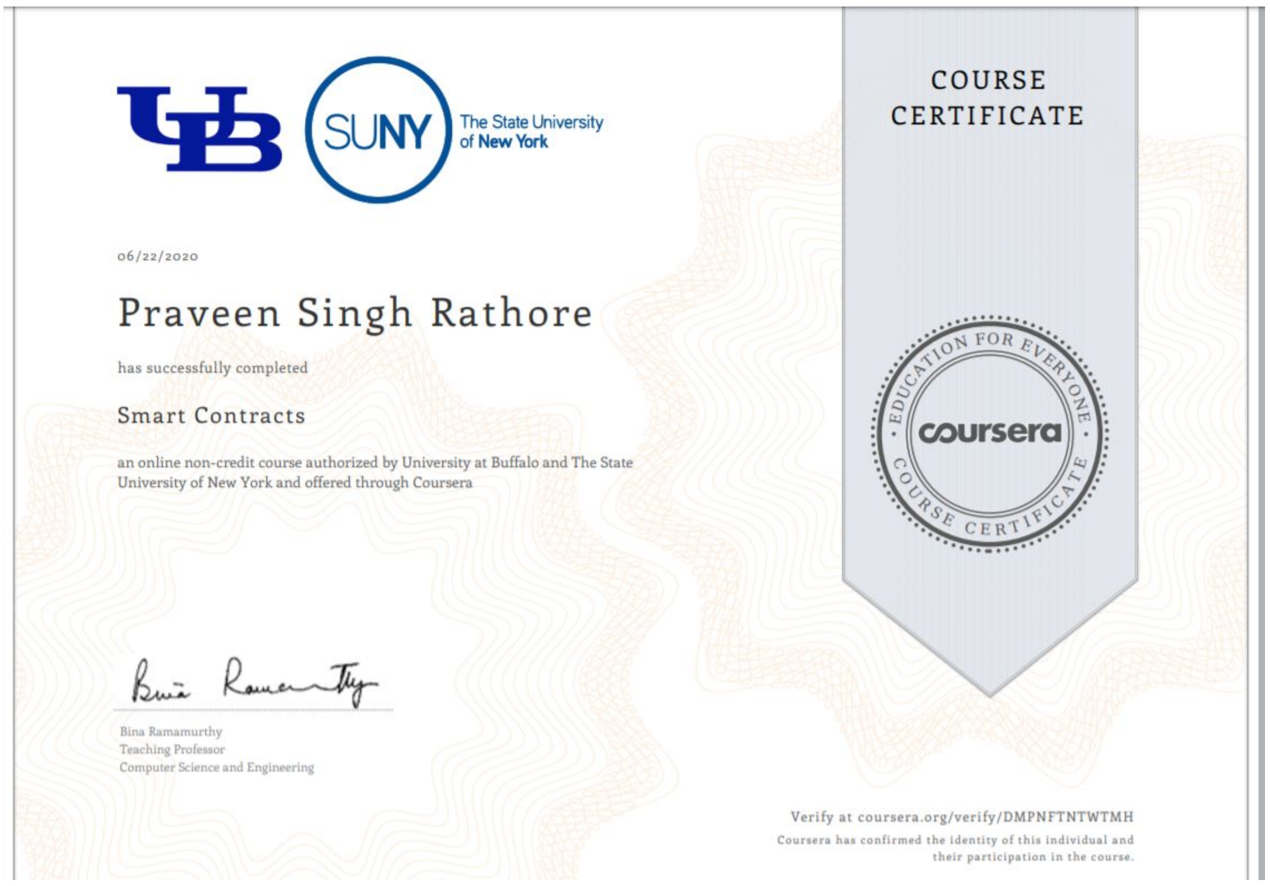
Last but not the least I thank the almighty for enlightening me with his blessings.

Praveen Singh Rathore

Enrollment Number: 20314802717

C-456/C-5

Certificate



COURSE CERTIFICATE

06/22/2020

Praveen Singh Rathore

has successfully completed

Smart Contracts

an online non-credit course authorized by University at Buffalo and The State University of New York and offered through Coursera

[illegible]

Bina Ramamurthy
Teaching Professor
Computer Science and Engineering

Verify at coursera.org/verify/DMPNFTNTWTMH
Coursera has confirmed the identity of this individual and their participation in the course.

ABSTRACT

A blockchain is a public ledger to which everyone has access but without a central authority having control. It is an enabling technology for individuals and companies to collaborate with trust and transparency. One of the best known applications of blockchains are the cryptographic currencies such as Bitcoin and others, but many other applications are possible. Blockchain technology is considered to be the driving force of the next fundamental revolution in information technology. Many implementations of blockchain technology are widely available today, each having its particular strength for a specific application domain.

A valid blockchain is a reliable way of confirming the party submitting a record to the blockchain, the time and date of its submission, and the contents of the record at the time of submission. A blockchain is an electronic ledger (register) of digital records, events, or transactions that are represented in condensed form known as a hash (digital security feature), authenticated, and maintained through a "distributed" or "shared" network of participants using a group consensus protocol (multiple users). Blockchain technology is already in use in the private sector, though clearly in the early stages of adoption, the most prevalent example being virtual currency known as Bitcoin.

Evolving networking scenarios include multi-administrative domain network services as drivers of novel business opportunities along with emerging operational challenges. As a potential approach to tackle upcoming requirements providing basic primitives to encompass analytics, automation, and distributed orchestration, we investigate blockchain-based decentralized applications (DApps) in the context of operational phases in support of multi-administrative domain networking.

This report covers the theory behind smart contract programming, with a special focus on Truffle, Solidity, and the Ethereum protocol. It gives a brief on Blockchain Technology and Decentralized Application including its history, nomenclature and few examples of recent apps. The chapter is laid out in a fashion that allows it to be used as a reference while creating a Decentralized Application from. This chapter is intended for readers who have prior exposure to programming.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1: Blockchain

The blockchain is an incorruptible digital ledger of economic transactions that can be programmed to record not just financial transactions but virtually everything of value." – Don & Alex Tapscott, authors Blockchain Revolution (2016).

A blockchain is, in the simplest of terms, a time-stamped series of immutable record of data that is managed by a cluster of computers not owned by any single entity. Each of these blocks of data (i.e. block) are secured and bound to each other using cryptographic principles (i.e. chain).

So, what is so special about it and why are we saying that it has industry disrupting capabilities?

The blockchain network has no central authority — it is the very definition of a democratized system. Since it is a shared and immutable ledger, the information in it is open for anyone and everyone to see. Hence, anything that is built on the blockchain is by its very nature transparent and everyone involved is accountable for their actions.

1.1 Blockchain Explained

A blockchain carries no transaction cost. (An infrastructure cost yes, but no transaction cost.) The blockchain is a simple yet ingenious way of passing information from A to B in a fully automated and safe manner. One party to a transaction initiates the process by creating a block. This block is verified by thousands, perhaps millions of computers distributed around the net. The verified block is added to a chain, which is stored across the net, creating not just a unique record, but a unique record with a unique history. Falsifying a single record would mean falsifying the entire chain in millions of instances. That is virtually impossible. Bitcoin uses this model for monetary transactions, but it can be deployed in many other ways.

1.2 The Three Pillars of Blockchain Technology

The three main properties of Blockchain Technology which has helped it gain widespread acclaim are as follows:

- Decentralization
- Transparency
- Immutability

Pillar #1: Decentralization

Before Bitcoin and BitTorrent came along, we were more used to centralized services. The idea is very simple. You have a centralized entity which stored all the data and you'd have to interact solely with this entity to get whatever information you required.

Another example of a centralized system is banks. They store all your money, and the only way that you can pay someone is by going through the bank.

The traditional client-server model is a perfect example of this:

When you google search for something, you send a query to the server who then gets back at you with the relevant information. That is simple client-server.

Now, centralized systems have treated us well for many years, however, they have several vulnerabilities.

- Firstly, because they are centralized, all the data is stored in one spot. This makes them easy target spots for potential hackers.
- If the centralized system were to go through a software upgrade, it would halt the entire system
- What if the centralized entity somehow shut down for whatever reason? That way nobody will be able to access the information that it possesses
- Worst case scenario, what if this entity gets corrupted and malicious? If that happens then all the data that is inside the blockchain will be compromised.

So, what happens if we just take this centralized entity away?

In a decentralized system, the information is not stored by one single entity. In fact, everyone in the network owns the information.

In a decentralized network, if you want to interact with your friend then you can do so directly without going through a third party. That was the main ideology behind Bitcoins. You and only you alone are in charge of your money. You can send your money to anyone you want without having to go through a bank.

FIgure1-1 Blockchain Basics

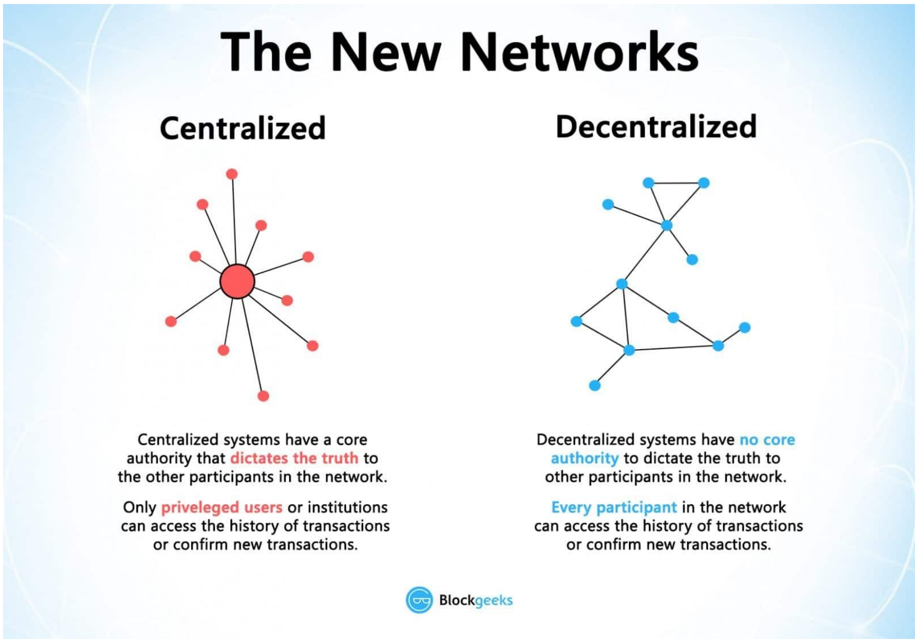

The New Networks
Centralized
Decentralized
Centralized systems have a core authority that dictates the truth to the other participants in the network.
Only priveleged users or institutions can access the history of transactions or confirm new transactions.
Decentralized systems have no core authority to dictate the truth to other participants in the network.
Every participant in the network can access the history of transactions or confirm new transactions.
Blockgeeks

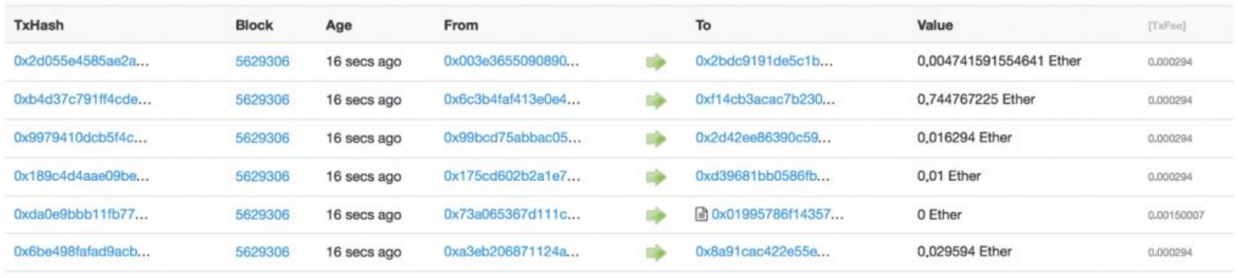
Figure1.2- Difference b/w Centralized and Decentralized

Pillar #2: Transparency

One of the most interesting and misunderstood concepts in blockchain technology is "transparency." Some people say that blockchain gives you privacy while some say that it is transparent. Why do you think that happens?

Well… a person's identity is hidden via complex cryptography and represented only by their public address. So, if you were to look up a person's transaction history, you will not see "Bob sent 1 BTC" instead you will see "1MF1bhsFLkBzzz9vpFYEmvwT2TbyCt7NZJ sent 1 BTC".

The following snapshot of Ethereum transactions will show you what we mean:



TxHash	Block	Age	From		To	Value	[TxFee]
0x2d055e4585ae2a...	5629306	16 secs ago	0x003e3655090890...	➡	0x2bdc9191de5c1b...	0,004741591554641 Ether	0.000294
0xb4d37c791ff4cde...	5629306	16 secs ago	0x6c3b4faf413e0e4...	➡	0xf14cb3acac7b230...	0,744767225 Ether	0.000294
0x9979410dcb5f4c...	5629306	16 secs ago	0x99bcd75abbac05...	➡	0x2d42ee86390c59...	0,016294 Ether	0.000294
0x189c4d4aae09be...	5629306	16 secs ago	0x175cd602b2a1e7...	➡	0xd39681bb0586fb...	0,01 Ether	0.000294
0xda0e9bbb11fb77...	5629306	16 secs ago	0x73a065367d111c...	➡	0x01995786f14357...	0 Ether	0.00150007
0x6be498fafad9acb...	5629306	16 secs ago	0xa3eb206871124a...	➡	0x8a91cac422e55e...	0,029594 Ether	0.000294

So, while the person's real identity is secure, you will still see all the transactions that were done by their public address. This level of transparency has never existed before within a financial system. It adds that extra, and much needed, level of accountability which is required by some of these biggest institutions.

Speaking purely from the point of view of cryptocurrency, if you know the public address of one of these big companies, you can simply pop it in an explorer and look at all the transactions that they have engaged in. This forces them to be honest, something that they have never had to deal with before.

However, that’s not the best use-case. We are pretty sure that most of these companies won’t transact using cryptocurrencies, and even if they do, they won’t do ALL their transactions using cryptocurrencies. However, what if the blockchain technology was integrated…say in their supply chain?

You can see why something like this can be very helpful for the finance industry right?

Pillar #3: Immutability

Immutability, in the context of the blockchain, means that once something has been entered into the blockchain, it cannot be tampered with.

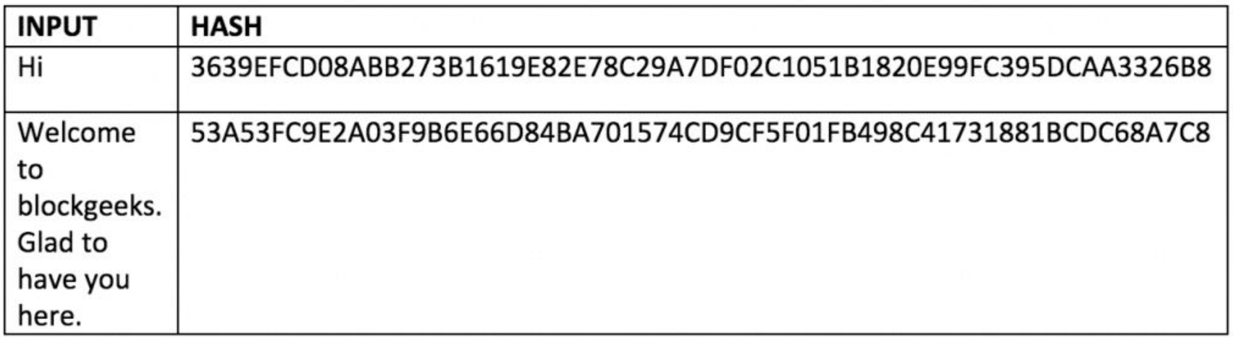
Can you imagine how valuable this will be for financial institutes?

Imagine how many embezzlement cases can be nipped in the bud if people know that they can’t “work the books” and fiddle around with company accounts.

The reason why the blockchain gets this property is that of cryptographic hash function.

In simple terms, hashing means taking an input string of any length and giving out an output of a fixed length. In the context of cryptocurrencies like Bitcoin, the transactions are taken as an input and run through a hashing algorithm (bitcoin uses SHA-256) which gives an output of a fixed length.

Let’s see how the hashing process works. We are going to put in certain inputs. For this exercise, we are going to use the SHA-256 (Secure Hashing Algorithm 256).



INPUT	**HASH**
Hi	3639EFCD08ABB273B1619E82E78C29A7DF02C1051B1820E99FC395DCAA3326B8
Welcome to blockgeeks. Glad to have you here.	53A53FC9E2A03F9B6E66D84BA701574CD9CF5F01FB498C41731881BCDC68A7C8

Figure1.3 Secure Hashing Algorithm 256(a)

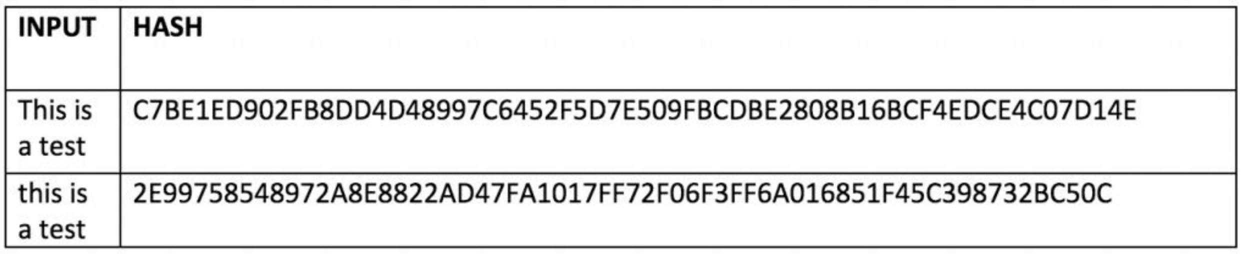
As you can see, in the case of SHA-256, no matter how big or small your input is, the output will always have a fixed 256-bits length. This becomes critical when you are dealing with a huge amount of data and transactions. So basically, instead of remembering the input data which could be huge, you can just remember the hash and keep track.

A cryptographic hash function is a special class of hash functions which has various properties making it ideal for cryptography. There are certain properties that a cryptographic hash function needs to have in order to be considered secure. You can read about those in detail in our guide on hashing.

There is just one property that we want you to focus on today. It is called the "Avalanche Effect."

What does that mean?

Even if you make a small change in your input, the changes that will be reflected in the hash will be huge. Let's test it out using SHA-256:



INPUT	HASH
This is a test	C7BE1ED902FB8DD4D48997C6452F5D7E509FBCDBE2808B16BCF4EDCE4C07D14E
this is a test	2E99758548972A8E8822AD47FA1017FF72F06F3FF6A016851F45C398732BC50C

Figure1.3 Secure Hashing Algorithm 256(b)

1.3 Types of Blockchain

There are basically two types of blockchains-

Private Blockchain like Ethereum and Bitcoin

Public Blockchain like Hyperledger and R3 Corda

1.3.1 What is Ethereum?

Ethereum is an open source distributed public blockchain network. It allows decentralized apps to be built on it with the help of Smart Contract functionality.

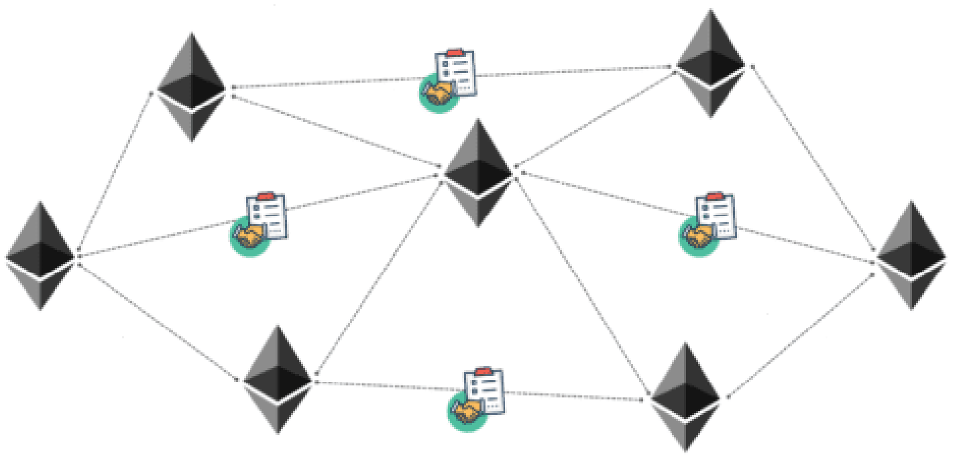


Figure1.4 Ethereum Network

Vitalik Buterin developed Ethereum as an extension to the original core blockchain concept. He improvised Bitcoin's protocols to support applications beyond currency issuance. Its major breakthrough is the ability to easily write and deploy Smart Contracts. These are actually bits of code that are executed on the network. Hence, this platform could help developers to write programs for building decentralized organisations.

Anyone across the globe can connect with Ethereum blockchain and can maintain the current state of the network. Therefore, Ethereum is also widely referred to as the "World Computer".

1.3.2 What is Hyperledger?

"Hyperledger is an open source development project to benefit an ecosystem of Hyperledger based solution providers and users. It is focused on blockchain related use cases that will work under a variety of industrial sectors." – Brian Behlendorf (Executive Director, Hyperledger)

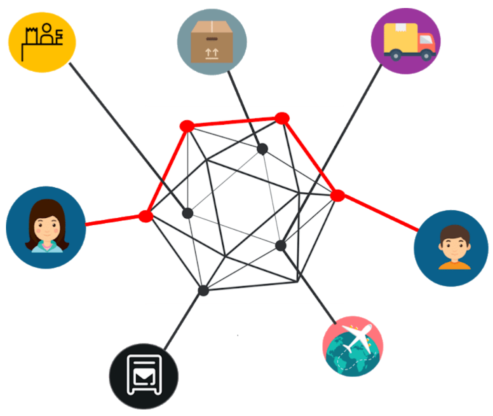


Figure1.5 Hyperledger

Every business and industry is distinctive in their own way and the applications serving to their needs must be personalized. The Ethereum Blockchain works with a very generalized protocol for everything that runs on its network. You can think of Hyperledger, on the other hand, as a software for people to develop their own personalized blockchains tending to the needs of their businesses.

Hyperledger is an open source collaborative project hosted by The Linux Foundation. It is neither a tool nor a platform like Ethereum. It's an umbrella strategy with multiple platforms for developing enterprise solutions.

1.3.4 Hyperledger vs Ethereum

Features	**Hyperledger**	**Ethereum**
Purpose	Preferred platform for B2B businesses	Platform for B2C businesses and generalized applications
Confidentiality	Confidential transactions	Transparent
Mode of Peer Participation	Private and Permissioned Network	Public/Private and Permissionless Network
Consensus Mechanism	Pluggable Consensus Algorithm: No mining required	PoW Algorithm: Consensus is reached by mining
Programming Language	Chaincode written in Golang	Smart Contracts written in Solidity
Cryptocurrency	No built-in cryptocurrency	Built-in cryptocurrency called Ether

Chapter 2: What Is a Decentralized Application?

A new model for building massively scalable and profitable applications is emerging. Bitcoin paved the way with its cryptographically stored ledger, scarce-asset model, and peer-to-peer technology. These features provide a starting point for building a new type of software called decentralized applications, or dapps. Dapps are just now gaining media coverage but will, I believe, someday become more widely used than the world's most popular web apps. They are more flexible, transparent, distributed, resilient, and have a better incentivized structure than current software models. This is the first book that will help you to understand them and create your own.

2.1 The emergence of Dapps

A new model for building successful and massively scalable applications is emerging. Bitcoin led the way with its open-source, peer-to-peer nature, cryptographically-stored records (block chain), and limited number of tokens that power the use of its features. In the last year dozens of applications are adopting the Bitcoin model in order to succeed. Ethereum, Omni and the SAFE Network are just a few of those "decentralized applications" that use a variety of methods to operate. Some use their own block chain (Ethereum), some use existing blockchains and issue their own tokens (Omni Layer), and others operate at two layers above an existing block chain and issue their own tokens (SAFE Network).

This paper describes why decentralized applications have the potential to be immensely successful, how the different types of decentralized applications can be classified, and introduces terminology that aims to be accurate and helpful to the community. Finally, this paper postulates that these decentralized applications will some day surpass the world's largest software corporations in utility, user-base, and network valuation due to their superior incentivization structure, flexibility, transparency, resiliency, and distributed nature.

2.2 What Is a Decentralized Application?

Most people are familiar with the term "application" as it pertains to software. A software application is software that defines a specific goal. There are millions of software applications currently in use, and the vast majority of web software applications follow a centralized server-client model. Some are distributed, and a select few novel ones are decentralized. Figure 1-1 shows a visual representation of these three models for software. Centralized systems are currently the most widespread model for software applications. Centralized systems directly control the operation of the individual units and flow of information from a single center. All individuals are directly dependent on the central power to send and receive information and to be commanded. Facebook, Amazon, Google, and every other mainstream service we use on the Internet uses this model. Let's call these huge services "The Stacks." The Stacks are useful because they provide a valuable service to us, but they have immense flaws.

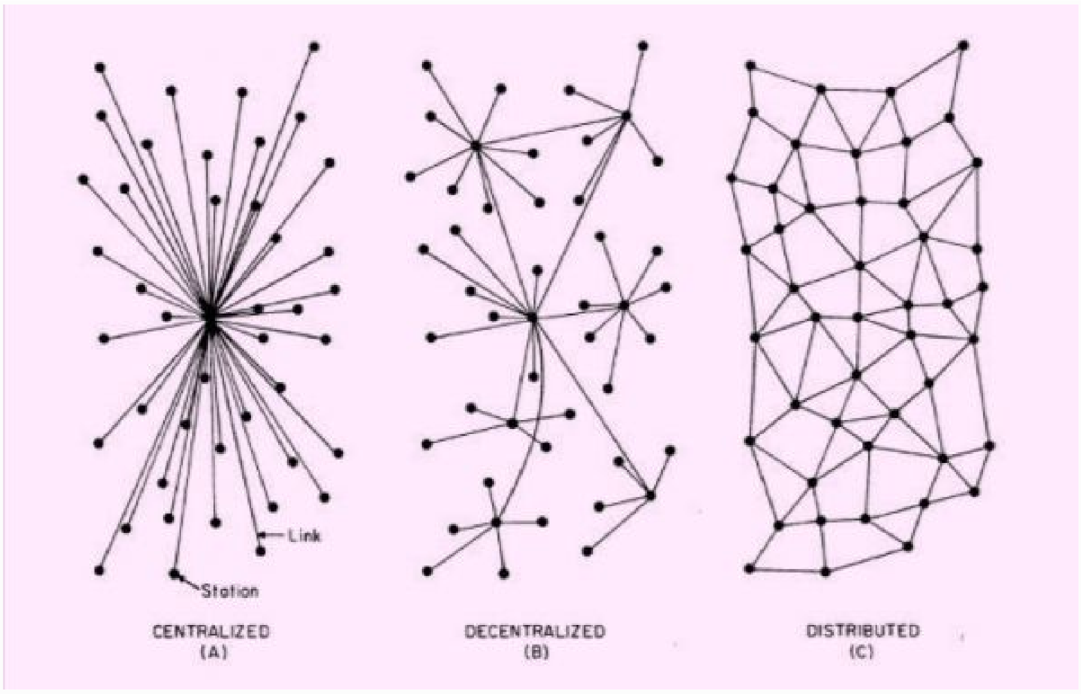


Figure 2-1. The three different types of software applications .

2.2.1 Definition of a Dapp

For an application to be considered a Dapp (pronounced Dee-app, similar to Email) it must meet the following criteria:

1. The application must be completely open-source, it must operate autonomously, and with no entity controlling the majority of its tokens. The application may adapt its protocol in response to proposed improvements and market feedback but all changes must be decided by consensus of its users.
2. The application's data and records of operation must be cryptographically stored in a public, decentralized blockchain in order to avoid any central points of failure.
3. The application must use a cryptographic token (bitcoin or a token native to its system) which is necessary for access to the application and any contribution of value from (miners / farmers) should be rewarded in the application's tokens.
4. The application must generate tokens according to a standard cryptographic algorithm acting as a proof of the value nodes are contributing to the application (Bitcoin uses the Proof of Work Algorithm).

So, what's the difference between decentralized and distributed?

Distributed means computation is spread across multiple nodes instead of just one. Decentralized means no node is instructing any other node as to what to do. A lot of Stacks such as Google have adopted a distributed architecture internally to speed up computing and data latency. This means that a system can be both centralized and distributed.

Can a system be both distributed and decentralized?

Yes, it can. Bitcoin is distributed because its time stamped public ledger, the block‐ chain, resides on multiple computers. It's also decentralized because if one node fails, the network is still able to operate. That means that any app that uses a blockchain alongside other peer-to-peer tools can be distributed and decentralized.

So, is having decentralized consensus the only requirement to being a decentralized app?

The dapp space is currently an emerging field with a lot of smart people still experimenting with new models. Different developers have different opinions on what exactly a dapp is. Some developers think that having no central point of failure is all it takes and some think that there are other requirements. The focus of this book is to talk about profitable dapps; that is, dapps from which developers and users can earn money. The reason for the profit focus is because profit is the cornerstone of a successful, robust, and sustainable dapp. Incentives keep developers building, users loyal, and miners maintaining a blockchain. To that end, Figure 1-2 shows the four features any profitable dapp should have.

Feature 1: Open Source

Decentralized, closed-source applications require users to trust that the app is as decentralized as the core developers say it is, and that they don't have access to their data through a central source. Closed-source applications thus raise a red flag to users and act as a barrier to adoption. The aversion to closed source is particularly pro‐ nounced when the application is designed to receive, hold, or transfer user funds. Although it might not be impossible to successfully launch a closed-source decentralized application, the battle would be uphill from the start, and users would favor open 4 | Chapter 1: What Is a Decentralized Application? source competitors. Open sourcing a dapp changes the structure of its business prac‐ tices so that the Internet is common denominator instead of a chain of closed silos.

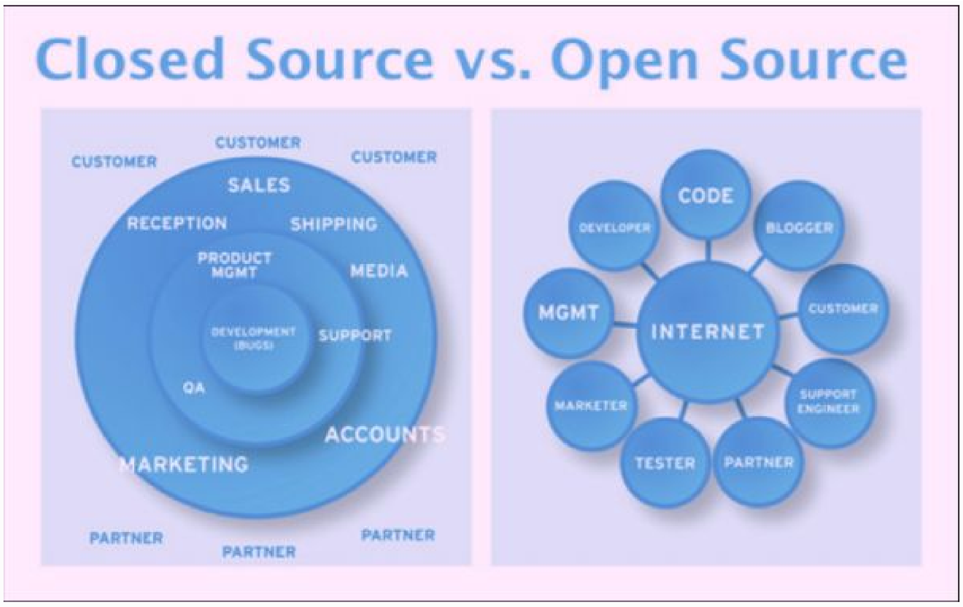


Figure 2-2. Closed source versus open source business plans

Feature 2: Internal Currency

A question that consistently comes up in dapp circles is how to monetize a dapp. Traditional modes of monetization for centralized applications include transaction fees, advertising revenues, referral commissions, access rights to user data, and subscrip‐ tion services. If you open source your dapp, how are you supposed to make money? You might try programmatically inserting a fee for transactions in the network that would automatically go to the app developers' account, but that would rely on trust‐ ing users to not fork the app and take out your commission—not ideal. Neither is embedding advertising, subscription services, or any of the other centralized business models.

Feature 3: Decentralized Consensus

Before Bitcoin, consensus on transaction validity always required some degree of centralization. If you wanted to make a payment, your transaction had to go through a clearinghouse that monitored all transactions. Bitcoin is peer to peer (P2P), which means nodes are able to talk to each other directly. P2P networks are not a novel thing; Distributed Hash Tables (DHTs) like BitTorrent were invented before the blockchain. DHTs are great for storing and streaming decentralized data, but if you want application-level constructs like usernames, status updates, high scores, and so forth for which you need everyone to agree on in a decentralized way, you'll also need a blockchain.

Feature 4: No Central Point of Failure

Dapps can't be shut down, because there is no server to take down. Data in a dapp is decentralized across all of its nodes. Each node is independent; if one fails, the others are still able to run on the network. There are a number of decentralized database systems on which to build Dapps that allow for this feature, such as Interplanetary File System, BitTorrent, and independent DHTs.

2.3 The History of Decentralized Applications

In its early days, the Web was obviously not as useful as it is today with the array of apps and services that do everything under the sun, but it did have a more DIY distributed feel to it. The Web was pretty decentralized from the outset. The HTTP protocol connected everyone on the planet with a computing device and an Internet connection. In the HTTP protocol guidelines, there are a set of trusted servers that translate the web address you enter into a server address. Furthermore, HTTPS adds another layer of trusted servers and certificate authorities. People would host personal servers for others to connect to, and everyone owned their data. But soon, application servers began taking off and the centralized model of data ownership as we know it today was born. Why did it happen this way?

The simple answer is because it was easy, both conceptually and programmatically. It was the easiest thing to do and it worked. One individual or group pays for maintenance of a server and profits from the users that utilize the software on it. Apps like MySpace and Yahoo! were among the first popular centralized apps. More recent apps like Uber and Airbnb decentralize the "real-world" parts of a business by providing a central and trusted data store. They are among the first to allow for participation in one moneymaking endeavor from all sides of the economy. Their decentralized busi- ness model foreshadows the development of even more decentralized apps.

As the HTTP web grew larger, a new protocol was introduced by a developer named Bram Cohen, called BitTorrent. BitTorrent is a protocol created as a solution to the lengthy time to download huge media files via HTTP and as an improvement on some of the P2P proposals before it, like Gnutella, Napster, and Grokster. The problem was that downloading huge files took a very long time and as the Web grew, so did the size of files that were available. Meanwhile, hard-drive space was increasing and more people were connected. BitTorrent solved this by making downloaders into uploaders, as well.
If there was a file you wanted, you would download it from not one, but multiple sources. The more popular the file, the more users who would be downloading it and subsequently uploading it, which meant you would be pulling from multiple sources. The more sources, the faster the download. Seeders were rewarded with faster down- load speeds, whereas leechers were punished with limited speeds. This tit-for-tat system of transferring data proved to be very useful for large media files like movies and TV shows.

2.4 Bitcoin as a Dapp

Satoshi Nakamoto, the creator of Bitcoin described his invention as "A Peer-to-Peer Electronic Cash System". Bitcoin has been shown to effectively solve the problems that arise from a trust-less and scalable electronic cash system by using a peer-to-peer, distributed ledger, the Bitcoin block chain. In addition to being a peer-to-peer electronic cash system however, Bitcoin is also an application that users can interact with through computer software). But most importantly for the purposes of this paper, based on the criteria outlined above, Bitcoin is a decentralized application. Here is why:

1. All Bitcoin software applications are open-source, no entity (government, company, or organization) controls Bitcoin and all records related to the use of Bitcoin are open and public.
2. Bitcoin generates its tokens, the bitcoins, with a predetermined algorithm that cannot be changed, and those tokens are necessary for Bitcoin to function. Bitcoin miners are rewarded with bitcoins for their contributions in securing the Bitcoin network.
3. All changes to Bitcoin must be approved by a majority consensus of its users through the proof-of-work mechanism.

2.5 Nomenclature and its importance

Decentralized applications were initially described as Decentralized Autonomous Corporations, DAC, in an article written by Daniel Larimer, of Invictus Innovations. This paper avoids the term corporation for two reasons.

First, because it carries with it unnecessary preconceptions. For instance, a corporation is established in a jurisdiction, it has shares, a CEO, employees, etc. Dapps, like Bitcoin, have none of these characteristics. In addition, the narrative is very important for the way Dapps are perceived by various nations and jurisdictions. The same way that governments struggle to learn and regulate Bitcoin because the concept of currency is associated with it, governments might be compelled to regulate an open-source computer program that is a decentralized application.

Second, because traditional corporations may engage in several techniques to raise capital (like selling shares of its stock and pay dividends or borrowing against its stock and pay interest) that a Dapp does not need. The concept of a Dapp is so powerful and elegant, because it does not include these traditional corporate techniques. The ownership of the Dapp's tokens is all that is required for the holder to use the system. It's that simple. The value of the tokens is determined by how much people value the application. All the incentives, all the monetization, all the ways to raise support are built into this beautifully simple structure. Dapps are not required to recreate the functions that used to be necessary in centralized corporations in order to balance the power of shareholders and offer returns for investors and employees.

2.6 Classification of Dapps

There are several characteristics according to which decentralized applications can be classified. we will classify Dapps based on whether they have their own block chain or they use the blockchain of another Dapp. Based on this criterion, there are three types of Dapps.

Type I decentralized applications have their own blockchain. Bitcoin is the most famous example of a type I decentralized application but Litecoin and other "alt-coins" are of the same type.

Type II decentralized applications use the blockchain of a type I decentralized application. Type

II decentralized applications are protocols and have tokens that are necessary for their function. The Omni Protocol is an example of a type II decentralized application.

Type III decentralized applications use the protocol of a type II decentralized application. Type III decentralized applications are protocols and have tokens that are necessary for their function. For example the SAFE Network that uses the Omni Protocol to issue ‘safecoins’ that can be used to acquire distributed file storage is an example of a type III decentralized applications.

A useful analogy for a type I Dapp is a computer operating system (like Windows, Mac OS X, Linux, Android, iOS) for a type II Dapp a general purpose software program (like a word processor, spreadsheet software, a file synchronization system such as Dropbox) and for type III Dapp, a specialized software solution (like a mail-merge tool that uses a word processor, an expense report macro that uses a spreadsheet, or a blogging platform that uses Dropbox.) Using this analogy, it may be expected that due to network effects and the ecosystem surrounding each decentralized application, there will be a few type I Dapps, more type II Dapps and even more type III Dapps.

At this point, it is important to mention that there are currently several excellent open-source projects that leverage type I Dapps. Colored coins and CoinJoin, for example, are based on the Bitcoin block chain and provide useful features to their users. These projects however cannot be classified as a type II Dapps, according to our definition, because they don’t issue and manage a token. (The development and operation of these projects depends on donations instead.)

Chapter3: Few Examples of recent Decentralized Apps

Let’s look through just a few examples of recent decentralized apps.

PopcornTime

PopcornTime uses the BitTorrent protocol to stream videos between users in real time, kind of like Netflix for torrents. It is the worst nightmare of the Motion Pic- ture Association of America (MPAA). No regulator can shut it down, and now everyone has access to free movies. PopcornTime proved to be a useful dapp acting as a decentralized version of Netflix. The creators claim that it has been downloaded in every single country, even the two without Internet. PopcornTime uses no internal currency and doesn’t need decentralized consensus, so it had no use for a blockchain. It simply streams movies and that proved to provide a lot of value.

OpenBazaar

OpenBazaar aims to be a decentralized version of Ebay. No middleman can tell sellers what they can and can’t sell or decide on the fees for using the service. It’s built on the BitTorrent protocol, but the problem is that the sellers must host their own stores. They need to have their own server and leave it on in order for users to be able to see their items. Ideally sellers could just upload their store data to the network, perhaps paying a small fee, without having to worry about it. This requires a decentralized system of incentivized storage miners, which we’ll cover in detail in Chapter 4. Open- Bazaar uses BitTorrent’s protocol for data transfer and Bitcoin as currency for trans- actions between sellers.

FireChat

FireChat emerged with a famous use case—the 2014 Hong Kong protests for democracy. China’s infamous “Great Firewall” is notorious for blocking IP addresses for content that it deems pro democracy or just not in its interest. The protesters feared the government would try to shut down access to various social networks to stop collaboration as is possible to do with the HTTP protocol. Instead, they used Fire- Chat, an app that used a new feature in iOS 7 called multipeer connectivity makes it possible for phones to connect to each other directly without a third party. Because it The History of Decentralized Applications | 9 had no central point of failure, the government would be forced to manually shut down every node, and thus the protestors were able to communicate with one another robustly.
Decentralized rebellion at its finest.

Lighthouse

It is a Bitcoin wallet embedded with a series of smart contracts. These smart contracts help pledge money to certain projects just like Kickstarter. When the project goal has been reached, it becomes possible to retrieve the funds out of the project backer’s Lighthouse wallet. Pledgers can undo pledges at any point without the involvement of the project creator. Light- house is a great example of using the existing Bitcoin infrastructure to build your dapp. It is just a UI with some Bitcoin smart contracts built in as a wallet. It works and it builds off Bitcoin's existing user base. It has decentralized consensus, it's open source, it has no central point of failure, but it doesn't issue its own currency; rather, it uses Bitcoins. It's a useful dapp but it's not profitable for the creator.

Gems

Gems is a social-messaging app that is trying to create a more fair business model than WhatsApp. Gems is issuing its own currency and letting advertisers pay users directly with it for their data rather than acting as the middleman who profits. Users can also earn gems by referring others to the network. Gems are a meta-coin built on Bitcoin that developers also receive for developing and maintaining the software. As the iGems user base grows, so does the value of the currency. Users are incentivized to grow the network and earn money just like the developers. You can think of Gems as shares in the dapp. Gems hasn't open sourced its code, so users can't verify if they truly have no central point of failure. It's a profitable app, but in my opinion it isn't robust enough to withstand competitors who fulfill the other three criteria.

So, are there any standalone dapps that satisfy all four criteria: no central point of failure, issue their own internal currency, have decentralized consensus, and are open source?

There are plenty of cryptocurrencies that satisfy all four criteria, but cryptocurrencies aren't dapps. I'm talking about decentralized social networks, ride sharing, search engines: taking The Stacks and decentralizing them. The answer is not yet. It's possible, though—the technology exists, and as soon as a few emerge, a flurry of developers will jump on the decentralized bandwagon to make some serious money for both themselves and their users.

Let's talk about some of these enabling technologies.

3.1 Enabling Technologies

I've already mentioned many of the enabling technologies during our discussion on the history of decentralized applications. Bitcoin's blockchain is, of course, of primary importance, so we'll take a deeper dive into this before considering the other enabling technologies. The blockchain helped solve the Byzantine Generals Problem. That problem asks the question, "How do you coordinate among distributed nodes to come up with some sort of consensus that is resistant to attackers trying to undermine it?" The proof-of-work algorithm and the blockchain help solve this.

When Bitcoin was created, decentralized consensus became possible. Proof-of-work isn't perfect—it is both computationally and energy expensive. There are alternative cryptocurrencies out there that solve meaningful problems, like PrimeCoin, whose miners use their compute resources to find prime numbers. In a world where Bitcoin is the de facto currency, we're going to be using a lot of energy to maintain the net- work, energy that could be put to better use than just helping the network maintain itself.

The problem is that proof-of-work is the only known Sybil-prevention system thus far. Consensus research is still ongoing and has not stopped with proof-of-work, but for now it’s the best that we have. In terms of up-and-coming competitors to proof-of-work, there is a big one that comes to mind: proof-of-stake.

Proof-of-stake isn’t perfect, either, but it can complement proof-of-work. Proof-of-stake is a consensus mechanism that relies instead on computational power to prevent Sybil attacks on stake in the network. Usually, by stake it means amount of cryptocurrency owned by the miner. The idea is that the more cryptocurrency you have, the more invested you are in ensuring the stability of the network and the less likely you are to perform a 51 percent attack to fork the blockchain. Delegated proof-of-stake is an innovation of proof-of-stake where a set of 101 delegates can vote on block generators. Both delegated proof-of-stake and proof-of-stake are still undergo- ing research, but if either proves to be secure in the long term, they could be used to complement or maybe even completely replace proof-of-work.

Chapter4: The Operation of a Dapp

4.1 Mechanisms for establishing consensus

There are two common mechanisms by which Dapps can establish consensus: the proof-of-work, POW, mechanism and the proof of stake, POS, mechanism.

With the proof-of-work mechanism, decisions about changes in a Dapp are made based on the amount of work that each stakeholder contributes to the operation of the Dapp. Bitcoin uses that approach for its day-to-day operation. The mechanism for establishing consensus through POW is commonly called mining.

With the proof-of-stake mechanism, decisions about changes in the Dapp are made based on the percent ownership that various stakeholders have over the application. For instance, the vote of a stakeholder who controls 10% of the tokens issued by a Dapp, carries a 10% weight. The Omni Protocol is based on the POS mechanism.

The two mechanisms can be used in parallel, as is the case with Peercoin. Such a combination allows a Dapp to operate with less energy consumption than proof-of-work alone, and allows it to be more resistant to 51% attacks.

4.2 Mechanisms for distributing tokens

There are three common mechanisms by which Dapps can distribute their tokens: mining, fund-raising and development.

With the mining mechanism, tokens are distributed to those who contribute most work to the operation of a Dapp. Taking Bitcoin as an example, bitcoins are distributed through a predetermined algorithm to the miners that verify transactions and maintain the Bitcoin block chain.

With the fund-raising mechanism, tokens are distributed to those who fund the initial development of the DApp. Taking the Master Protocol as an example, Mastercoins were initially distributed to those who sent bitcoins to a given address at the rate of 100 Mastercoins per bitcoin sent. The bitcoins collected were then used to fund the development of applications that promoted the development of the Master Protocol.

With the development mechanism, tokens are generated using a predefined mechanism and are only available for the development of the DA. For example, in addition to its fund-raising mechanism, the Master Protocol used the collaboration mechanism to fund its future development. An additional 10% of the Mastercoins generated through fund-raising was set aside for development of the Master Protocol. Those Mastercoins become available through a predetermined schedule and are distributed via a community-driven bounty system where decisions are made based on the proof-of-stake mechanism.

To summarize: Tokens of a Dapp that establishes consensus through proof-of-work are distributed by mining, by people buying directly from miners and by trading for goods and services; that is the case with Bitcoin. Tokens of a Dapp that establishes consensus through proof-of-stake are distributed based on the contribution of stakeholders during a fundraiser, by people collaborating on the development of the Dapp and by trading for goods and services; that is the case with the Omni Protocol.

4.3 Formation and development of a Dapp

Development of decentralized applications takes place in three steps.

Step 1: A whitepaper is published describing the Dapp and its features

As in the case of Bitcoin, the most common way by which a Dapp takes form is by the public release of a whitepaper that describes the protocol, its features, and its implementation. After the public release, feedback from the community is necessary for the further development of the DA.

Step 2: Initial tokens are distributed

If the Dapp is using the mining mechanism to distribute its tokens, a reference software program is released so that it can be used for mining. In the case of Bitcoin, a reference software program was released and the initial transaction block was created.

If the Dapp is using the fund-raising mechanism, a wallet software becomes available to the stakeholders of the Dapp, so that they can exchange the tokens of the DA. In the case of Mastercoin, an Exodus fund-raising address and a wallet script were publicly released.

If the Dapp is using the development mechanism, a bounty system is put in place that allows the suggestion of tasks to be performed, the tracking of the people who are working on those tasks and the criteria by which bounties can be awarded.

Step 3: The ownership stake of the Dapp is spread

As tokens from mining, fund-raising and collaboration are distributed to a greater number of participants, the ownership of the Dapp becomes less and less centralized and participants that held a majority stake at earlier have less and less control. As the Dapp matures, participants with more diverse skills are incentivized to make valuable contributions, and the ownership of the Dapp is distributed further. Through market forces the tokens of a Dapp are transferred to those who value it the most. Those individuals then can contribute to the development of the Dapp in the areas that they have an expertise.

The case of Bitcoin illustrates the point. By some estimates, Satoshi Nakamoto mined many of the first 1,000,000 bitcoins. As developers contributed code to Bitcoin and miners contributed computational power to the Bitcoin network, the market began to value bitcoins more highly. As the system matured even more, people with diverse skills started valuing Bitcoin and contributing to its development. Now that more than 12 million bitcoins are in circulation and Satoshi Nakamoto's high original ownership stake has been diluted.

Chapter5: Smart Contracts

Smart contracts are a term that was coined by Nick Szabo in 1994. Smart contracts and blockchain are closely linked – smart contracts are the programs that we write on the blockchain and let us interact with it. It is in a smart contract that we can define business logic and let us code the rules with which we want to interact with the blockchain. The most powerful feature of a smart contract is the fact that once it is deployed on to the blockchain, it is immutable and you cannot go back and edit the programs. This makes us embrace a paradigm where we a piece of code with the logic that we want in it will be immutable and can execute by itself when it is called. It is smart contracts that truly unleash the power of the blockchain and hence it is important to learn to develop smart contracts on the blockchain.

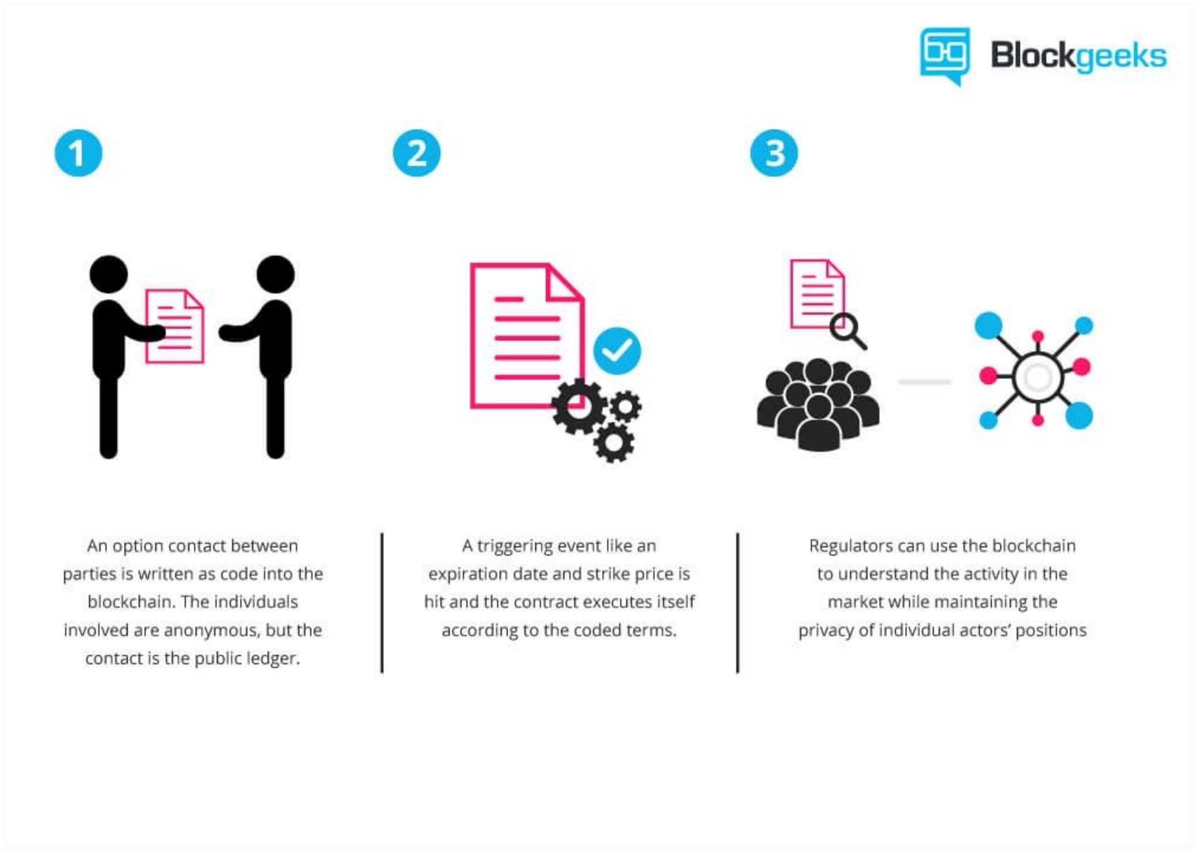
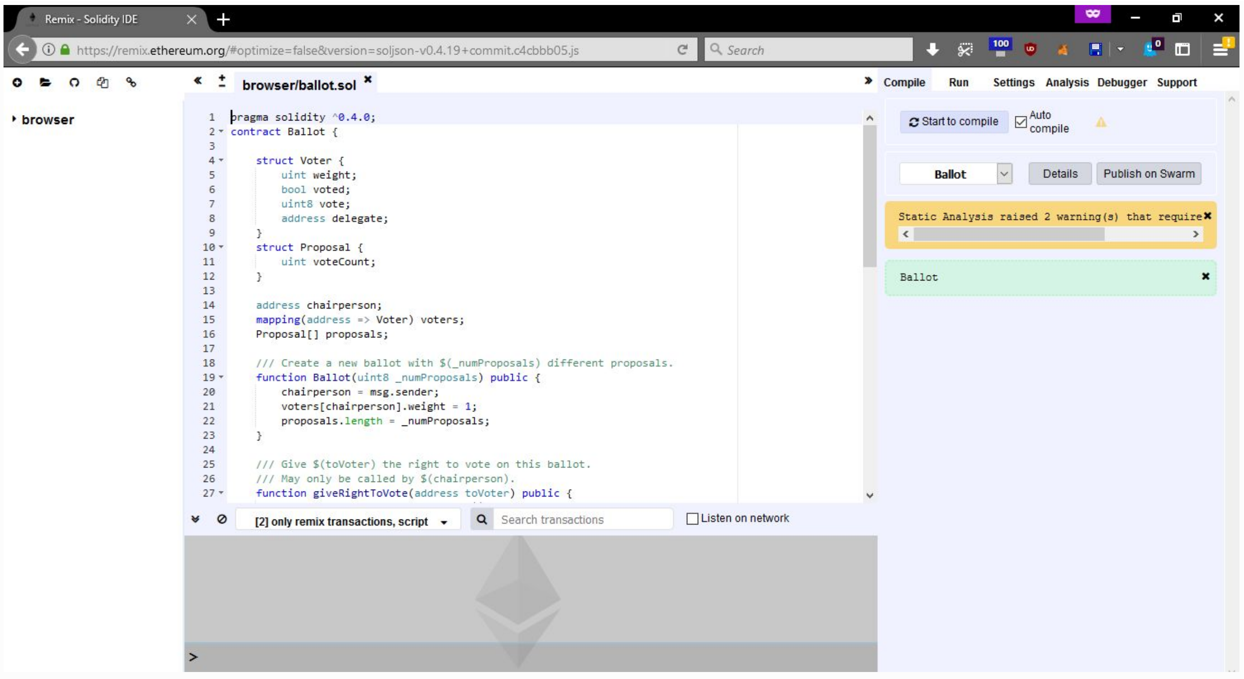

Blockgeeks
1
2
3
An option contact between parties is written as code into the blockchain. The individuals involved are anonymous, but the contact is the public ledger.
A triggering event like an expiration date and strike price is hit and the contract executes itself according to the coded terms.
Regulators can use the blockchain to understand the activity in the market while maintaining the privacy of individual actors' positions

Figure5.1 Working of a Smart Contracts

5.1 Get started with Solidity and Smart contracts

Image Credits: https://solidity.readthedocs.io/en/develop/

The ethereum foundation has released a cloud based IDE called Remix that is a simple way to get started. Simply go to http://remix.ethereum.org and you should be greeted with the following screen:



```
pragma solidity ^0.4.0;
contract Ballot {

    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
    }

    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;

    /// Create a new ballot with $(_numProposals) different proposals.
    function Ballot(uint8 _numProposals) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        proposals.length = _numProposals;
    }

    /// Give $(toVoter) the right to vote on this ballot.
    /// May only be called by $(chairperson).
    function giveRightToVote(address toVoter) public {
```

```
Static Analysis raised 2 warning(s) that require
```

```
Ballot
```

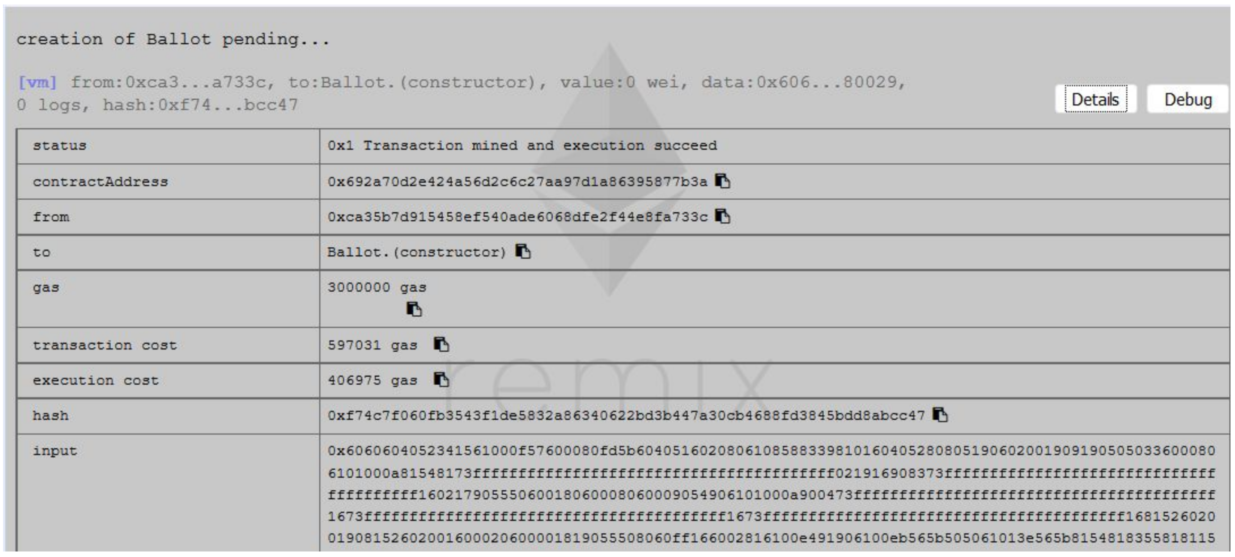
Figure5.2 The Remix default screen.

You see a screen on the left that lets you write code. The right hand side lets you deploy your code on the blockchain and lets you interact with the functions that you have written.

The simplest thing to do in any programming language is declare and write variables so let's see an example of how to do that:

```
pragma solidity ^0.4.0;

contract SimpleStorage {

  uint storedData;

  function set(uint x) {

      storedData = x;

  }

  function get() constant returns (uint retVal) {

      return storedData;

  }

}
```

Copy paste the following code on the code section in remix and then click on the deploy tab and click on create.
There are two functions – one that lets us set the variable value and one that lets us read it.
The first line of code tells us that we are using a particular version of Solidity. Contract SimpleStorage tells us the name of the contract and it helps us group code and logic into a single unit that can be referenced directly.
The next line of the code declares a variable – storeData is of type uint.The next two functions are simply getter and setter functions of the variable storeData.
Let’s take a minute to see what happens when we click on create. Every time you click on create, you are taking your Solidity based smart contract and deploying it on the blockchain. However, to simplify things and make it faster to develop applications, the blockchain that it deploys to is the ethereum virtual machine and resides in your browser and lets you interact with it. Since code once deployed cannot be changed, each time you have to make a change to your code, you have to redeploy it and test it out again.



```
creation of Ballot pending...

[vm] from:0xca3...a733c, to:Ballot.(constructor), value:0 wei, data:0x606...80029,
0 logs, hash:0xf74...bcc47
```

status	0x1 Transaction mined and execution succeed
contractAddress	0x692a70d2e424a56d2c6c27aa97d1a86395877b3a
from	0xca35b7d915458ef540ade6068dfe2f44e8fa733c
to	Ballot.(constructor)
gas	3000000 gas
transaction cost	597031 gas
execution cost	406975 gas
hash	0xf74c7f060fb3543f1de5832a86340622bd3b447a30cb4688fd3845bdd8abcc47
input	0x6060604052341561000f57600080fd5b6040516020806108588339810160405280805190602001909190505033600080 6101000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffff ffffffffff16021790555060018060008060009054906101000a900473ffffffffffffffffffffffffffffffffffffffff 1673ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff1681526020 01908152602001600020600001819055508060ff166002816100e491906100eb565b505061013e565b8154818355818115

Figure5.3 The console output that you see when you create the smart contract.

All you need to do is to deploy to the mainnet blockchain (the actual ethereum blockchain) is change the environment in the run tab and you should be able to write to the blockchain – keep in mind that interacting with the ethereum main-net will actually cost you money and every time you call a function or deploy a new contract, you have to pay a gas cost in ethereum to make sure that your contract or function call gets mined and that you get your output.
Once you are up and running with Ethereum, you can read up the Solidity documentation to get yourself familiarized with the platform and its syntax. Solidity is fairly easy to learn and is very similar to javascript in its syntax.
First, let us take this opportunity to also see the various datatypes that solidity gives us:

1. **Uint** – unsigned int that can take only positive values.
2. **Address** – a type that is created in ethereum that can store wallet addresses.
3. **Mapping** – mappings are very useful to keep track of key value pairs. They are declared as mapping(_KeyType => _ValueType). If we make the KeyType as an address, we can keep track of certain information against a particular address.
4. **Bool** – a boolean variable that can be ether true or false.

Let us also take time here to talk about **Events** and what they mean in solidity. Events let other programs (or even your own program) listen to this particular event on the blockchain and then react after it is executed. As soon as the event is fired, all the parameters that are called with the event are passed on the code that is listening to it and can do some additional operations on the code. This is very useful for Javascript callbacks that might be interacting with our Dapp.

5.2 Web3

Web3 is the javascript library that we can include in our project to help us talk to the underlying contract and make calls to and from the ethereum blockchain. The web3 library is what interfaces between our web application and our server side code that is written on the blockchain via smart contracts.

Figure5.4 WEB3 Logo

5.2.1 How to integrate with Web3

You can install Web3 either via npm, meteor or just by including the javascript library that can be downloaded.

TestRPC.

A simple way to get started with interacting with the blockchain through our smart contract is to have a local instance of the blockchain running in your system. This way you can develop and checking things quite quickly and don't need to worry about things like block confirmation timings. This local instance of the blockchain is called TestRPC.

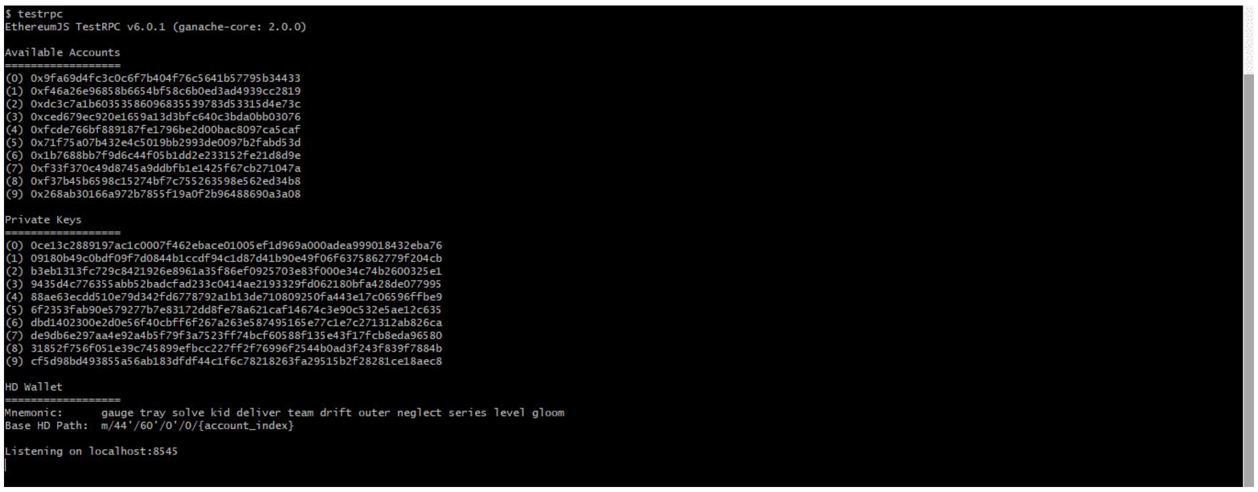
The easiest way to get started with TestRPC is via npm. If you don't know what npm is or don't have it installed, you should install node.js for your operating system. Then in the command line type in:

https://www.npmjs.com/package/ethereumjs-testrpc

Once the command finishes, you can start testrpc by running the command:

testrpc

This is what your screen should look like after you have run this command:



```
$ testrpc
EthereumJS TestRPC v6.0.1 (ganache-core: 2.0.0)

Available Accounts
==================
(0) 0x9fa69d4fc3c0c6f7b404f76c5641b57795b34433
(1) 0xf46a26e96858b6654bf58c6b0ed3ad4939cc2819
(2) 0xdc3c7a1b603535860968355 39783d53315d4e73c
(3) 0xced679ec920e1659a13d3bfc640c3bda0bb03076
(4) 0xfcde766bf889187fe1796be2d00bac8097ca5caf
(5) 0x71f75a07b432e4c5019bb2993de0097b2fabd53d
(6) 0x1b7688bb7f9d6c44f05b1dd2e233152fe21d8d9e
(7) 0xf33f370c49d8745a9ddbfb1e1425f67cb271047a
(8) 0xf37b45b6598c15274bf7c755263598e562ed34b8
(9) 0x268ab30166a972b7855f19a0f2b96488690a3a08

Private Keys
==================
(0) 0ce13c2889197ac1c0007f462ebace01005ef1d969a000adea999018432eba76
(1) 09180b49c0bdf09f7d0844b1ccdf94c1d87d41b90e49f06f6375862779f204cb
(2) b3eb1313fc729c8421926e8961a35f86ef0925703e83f000e34c74b2600325e1
(3) 9435d4c776355abb52badcfad233c0414ae2193329fd062180bfa428de077995
(4) 88ae63ecdd510e79d342fd6778792a1b13de710809250fa443e17c06596ffbe9
(5) 6f2353fab90e579277b7e83172dd8fe78a621caf14674c3e90c532e5ae12c635
(6) dbd1402300e2d0e56f40cbff6f267a263e587495165e77c1e7c271312ab826ca
(7) de9db6e297aa4e92a4b5f79f3a7523ff74bcf60588f135e43f17fcb8eda96580
(8) 31852f756f051e39c745899efbcc227ff2f76996f2544b0ad3f243f839f7884b
(9) cf5d98bd493855a56ab183dfdf44c1f6c78218263fa29515b2f28281ce18aec8

HD Wallet
==================
Mnemonic:      gauge tray solve kid deliver team drift outer neglect series level gloom
Base HD Path:  m/44'/60'/0'/0/{account_index}

Listening on localhost:8545
```

Figure5.5 Running TestRPC on your computer

TestRPC not only sets up a private instance of a blockchain, but it also creates 10 default accounts for you and each of those have a balance of 100 ETH in them. As you can see, it is quite favourable for testing out and does all the heavy lifting for you.

You can also connect your remix to the TestRPC so that your contracts are deployed on TestRPC instead of the javascript VM that remix provides.

The way to do that is to go to remix and then click on the Run tab on the right hand side. Then in the environment drop down click on Web3 Provider. It will then ask you if you want to connect to a node, after which it will show you localhost:8545 just click okay and you should be connected. If you got an error in this place, make sure that your URL is not https but http.

One that is done, you can click on the create button and your contract should be deployed on testRPC on your local system. Next, go to the complier tab and click on details and in the popup that comes, click on the copy button on the ABI section. Now we have the ABI for the contract with us. We also should copy the contract address and us that in the javascript code that is given below.

5.3 What is ABI ?

The best way to understand what an ABI is, is to compare with an API. Just like an API gives us the specifications that are needed to interact with an endpoint on the web, similarly, an ABI tells us the specifications that are needed to interact with the smart contract. It tells us of the various ways in which this contract can be called from an external source (like web3 in our case)

It is a combination of knowing the ABI and knowing which address the contract is deployed that we can call our solidity powered smart contracts from a web interface.

if(typeof web3 != 'undefined'){

console.log("Using web3 detected from external source like Metamask")

```
this.web3 = new Web3(web3.currentProvider)

}else{

this.web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8545"))

}

const MyContract = web3.eth.contract([here goes the ABI interface])

this.state.ContractInstance = MyContract.at("0x925d81c01d878899adbb7d38f84ce9d5284fa2e7")

}
```

The above piece of code lets you connect to the contract address and gives you an object through which you can interact with the contract and call a specific function.

```
yourContractInstance.bet(7, {

gas: 300000,

from: web3.eth.accounts[0],

value: web3.toWei(0.1, 'ether')

}, (err, result) => {

// Result is the transaction address of that function

})
```

Here we are calling our contract function with the value 7 and by setting certain default parameters that are needed.

We can call other methods of the contract as well and it is through this that we have now connected with the TestRPC and are able to make calls to the contract via the Web3 API.

Now we can tie this up with the usual javascript or jquery events that we are used to. We can trigger to call the function on the click on the button or on some other event. What is important to note is that the only piece of backend code that our app is talking to is the Solidity based smart contracts hosted on our testRPC.

Now that we have replaced the computing part of our Dapp we have taken the first major step towards building our own Dapp. Even if you are not particular about the other parts of a Dapp, you can actually host this online itself and it would be your first Dapp that is available to the public. The only major change that you will need to make before deploying this to point the web3 towards either of public test networks (Kovan Or Ropsten) and these you would have the re-deploy the contract on the new network.

Chapter6: Developing Ethereum DApps with Truffle

6.1 What Is Truffle Suite?

Truffle Suite is a development environment based on Ethereum Blockchain, used to develop DApps (Distributed Applications). Truffle is a one-stop solution for building DApps: Compiling Contracts, Deploying Contracts, Injecting it into a web app, Creating front-end for DApps and Testing.

Figure 6-1. Truffle Suite – Truffle Ethereum Tutorial

Truffle Suite has three components:

1. ***Truffle*** : It is a Development Environment, Testing Framework and Asset pipeline for Ethereum Blokchains
2. ***Ganache*** : Ganache is a personal Ethereum Blockchain used to test smart contracts where you can deploy contracts, develop applications, run tests and perform other tasks without any cost
3. ***Drizzle*** : Drizzle is a collection of libraries used to create easy and better front-end for Ethereum DApps.

Features Of Truffle Ethereum

Here's a list of features that makes Truffle a powerful tool to build Ethereum based DApps:

- Built-in support to Compile, Deploy and Link smart contracts
- Automated Contract testing
- Supports Console apps as well as Web apps
- Network Management and Package Management
- Truffle console to directly communicate with smart contracts

- Supports tight integration

6.2 What Is MetaMask?

MetaMask is an easy-to-use browser plugin (for Google-Chrome, Firefox and Brave browser), that provides a graphical user interface to make Ethereum transactions. It allows you to run Ethereum DApps on your browser without running a full Ethereum node on your system. Basically, MetaMask acts as a bridge between Ethereum Blockchain and the browser. MetaMask is open-source and provides the following exciting features:

- You can change the code of MetaMask to make it what you want it to be
- Provides built-in coin purchasing
- Local-key Storage

Figure 6-2. Truffle MetaMask – Truffle Ethereum Tutorial

Now, that we know about Truffle and MetaMask, let's get to the hands-on part of how to use these for DApps.

Installing Truffle and Creating a Truffle Project on Ubuntu

In this section of Truffle Ethereum tutorial, we will see how to install Truffle and how to create a Truffle project.

To install Truffle, you will have to run a simple command as below:

```
$ npm install -g truffle
```

Now, let's get to creating a project in Truffle. First, let us create a new directory and get into that directory using the following command:

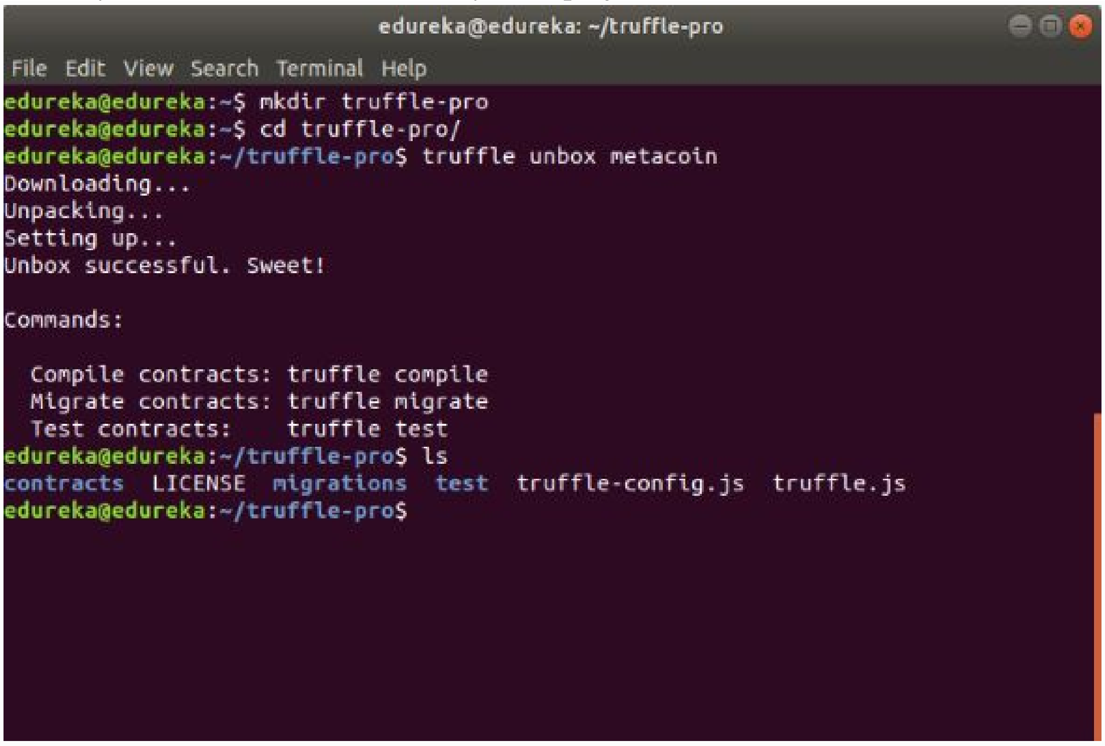
```
$ mkdir truffle-pro
```

$ cd truffle-pro

To create a project, execute the following command:

$ truffle unbox metacoin

When this command is successfully executed, you will see a project structure present in that directory with minimal files necessary for a project.


edureka@edureka: ~/truffle-pro
File Edit View Search Terminal Help

```
edureka@edureka:~$ mkdir truffle-pro
edureka@edureka:~$ cd truffle-pro/
edureka@edureka:~/truffle-pro$ truffle unbox metacoin
Downloading...
Unpacking...
Setting up...
Unbox successful. Sweet!

Commands:

  Compile contracts: truffle compile
  Migrate contracts: truffle migrate
  Test contracts:    truffle test
edureka@edureka:~/truffle-pro$ ls
contracts  LICENSE  migrations  test  truffle-config.js  truffle.js
edureka@edureka:~/truffle-pro$
```

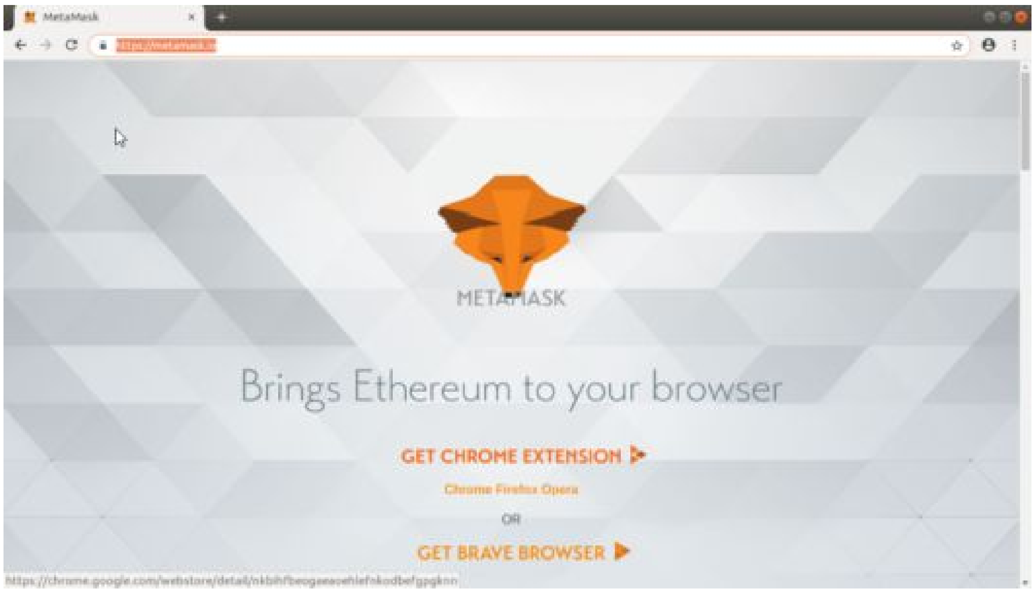
That's it! You have created a simple Truffle Ethereum project.

6.2.1 Installing MetaMask On Google Chrome

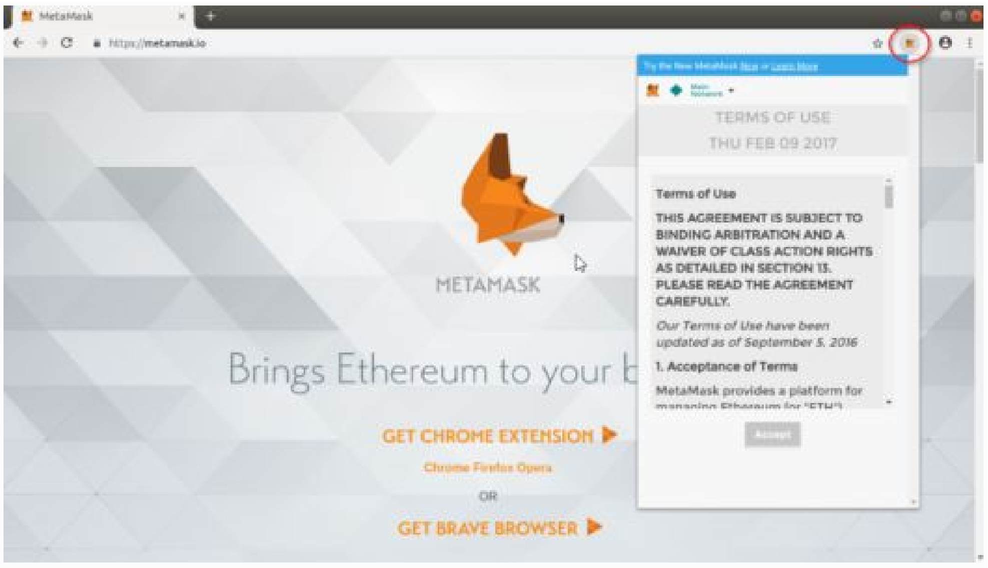
In this section of Truffle Ethereum tutorial, we will look at how to install MetaMask plugin for Google-Chrome browser.

Here are the steps to install MetaMask browser plugin:

1. First go to the following link: https://metamask.io/
2. Click on "GET CHROME EXTENSION" button. This will open a new tab


MetaMask
https://metamask.io
METAMASK
Brings Ethereum to your browser
GET CHROME EXTENSION
Chrome Firefox Opera
OR
GET BRAVE BROWSER
https://chrome.google.com/webstore/detail/nkbihfbeogaeaoehlefnkodbefgpgknn

3. Click on the “Add to Chrome” button and then “Add Extension“.
4. Now, on the Top-right corner of your browser, you can see the MetaMask icon.



Terms of Use

THIS AGREEMENT IS SUBJECT TO BINDING ARBITRATION AND A WAIVER OF CLASS ACTION RIGHTS AS DETAILED IN SECTION 13. PLEASE READ THE AGREEMENT CAREFULLY.

Our Terms of Use have been updated as of September 5, 2016

1. Acceptance of Terms

MetaMask provides a platform for

5. Accept the Terms and Conditions.

Now that we have Truffle Ethereum and MetaMask installed in the system, let’s see how we can develop a DApp using Truffle Ethereum and make transactions using MetaMask.

6.2.2 Installing TestRPC On Ubuntu

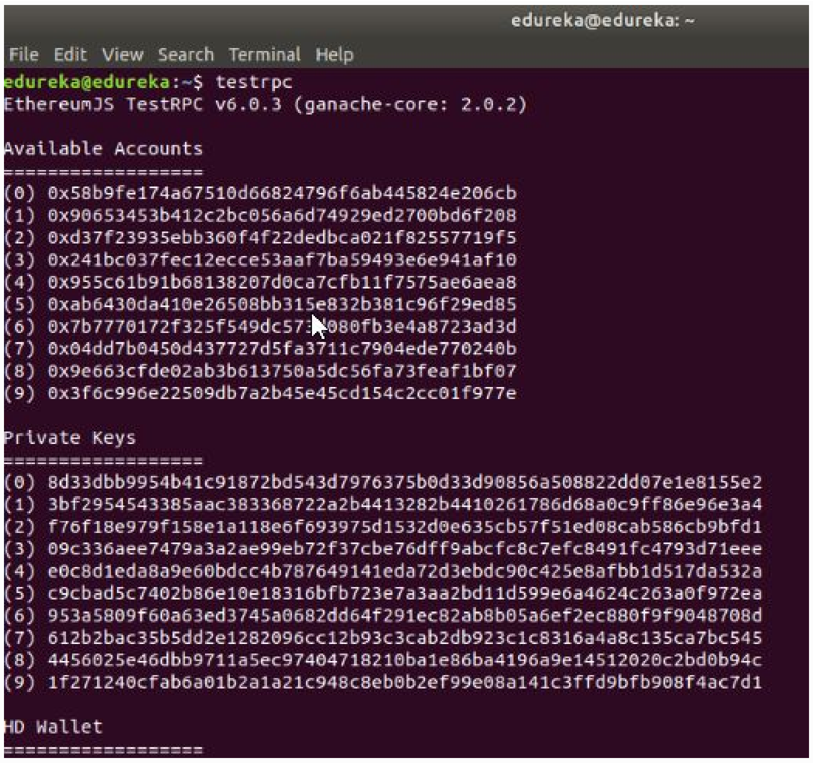
For this Truffle Ethereum tutorial, we will use “TestRPC”, which is a Blockchain emulator, to develop our DApp. TestRPC allows you to run a network for testing. It allows you to make calls to the Blockchain without running an actual Ethereum node.

To install TestRPC, run the following command:
$ npm install -g ethereumjs-testrpc

6.3 Demo: Developing A Simple DApp With Truffle And MetaMask And Making A Transaction

Open a new terminal and run TestRPC with the following command. This will start a test network on your system.
$ testrpc

You will see a list of available accounts, private keys for these accounts, a mnemonic phrase, and the port on which TestRPC is listening.



```
edureka@edureka:~$ testrpc
EthereumJS TestRPC v6.0.3 (ganache-core: 2.0.2)

Available Accounts
==================
(0) 0x58b9fe174a67510d66824796f6ab445824e206cb
(1) 0x90653453b412c2bc056a6d74929ed2700bd6f208
(2) 0xd37f23935ebb360f4f22dedbca021f82557719f5
(3) 0x241bc037fec12ecce53aaf7ba59493e6e941af10
(4) 0x955c61b91b68138207d0ca7cfb11f7575ae6aea8
(5) 0xab6430da410e26508bb315e832b381c96f29ed85
(6) 0x7b7770172f325f549dc57[illegible]080fb3e4a8723ad3d
(7) 0x04dd7b0450d437727d5fa3711c7904ede770240b
(8) 0x9e663cfde02ab3b613750a5dc56fa73feaf1bf07
(9) 0x3f6c996e22509db7a2b45e45cd154c2cc01f977e

Private Keys
==================
(0) 8d33dbb9954b41c91872bd543d7976375b0d33d90856a508822dd07e1e8155e2
(1) 3bf2954543385aac383368722a2b4413282b4410261786d68a0c9ff86e96e3a4
(2) f76f18e979f158e1a118e6f693975d1532d0e635cb57f51ed08cab586cb9bfd1
(3) 09c336aee7479a3a2ae99eb72f37cbe76dff9abcfc8c7efc8491fc4793d71eee
(4) e0c8d1eda8a9e60bdcc4b787649141eda72d3ebdc90c425e8afbb1d517da532a
(5) c9cbad5c7402b86e10e18316bfb723e7a3aa2bd11d599e6a4624c263a0f972ea
(6) 953a5809f60a63ed3745a0682dd64f291ec82ab8b05a6ef2ec880f9f9048708d
(7) 612b2bac35b5dd2e1282096cc12b93c3cab2db923c1c8316a4a8c135ca7bc545
(8) 4456025e46dbb9711a5ec97404718210ba1e86ba4196a9e14512020c2bd0b94c
(9) 1f271240cfab6a01b2a1a21c948c8eb0b2ef99e08a141c3ffd9bfb908f4ac7d1

HD Wallet
==================
```

Figure6.3 Test Blockchain server.

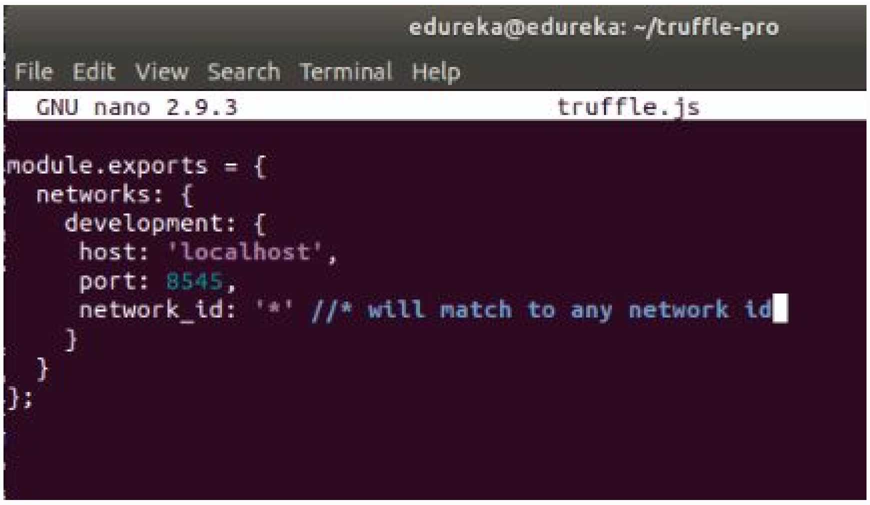
Note: Do not use the mnemonic phrase on the main Ethereum Network. Use it only on a private network.

Now, let’s setup truffle.

Open a new terminal and go into the directory where the project was created.

To run truffle on our network, we need to edit the “truffle.js” file. Open this file and make the following entries:

```
module.exports = {
    networks: {
        development: {
            host: 'localhost',
            port: 8545,
            network_id: '*' //* will match to any network id
            }
    } };
```



edureka@edureka: ~/truffle-pro
File Edit View Search Terminal Help
GNU nano 2.9.3
truffle.js

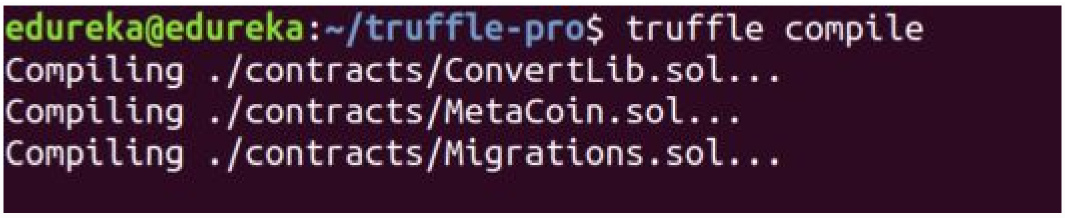
```
module.exports = {
  networks: {
    development: {
     host: 'localhost',
     port: 8545,
     network_id: '*' //* will match to any network id
    }
  }
};
```

Save the file and exit.

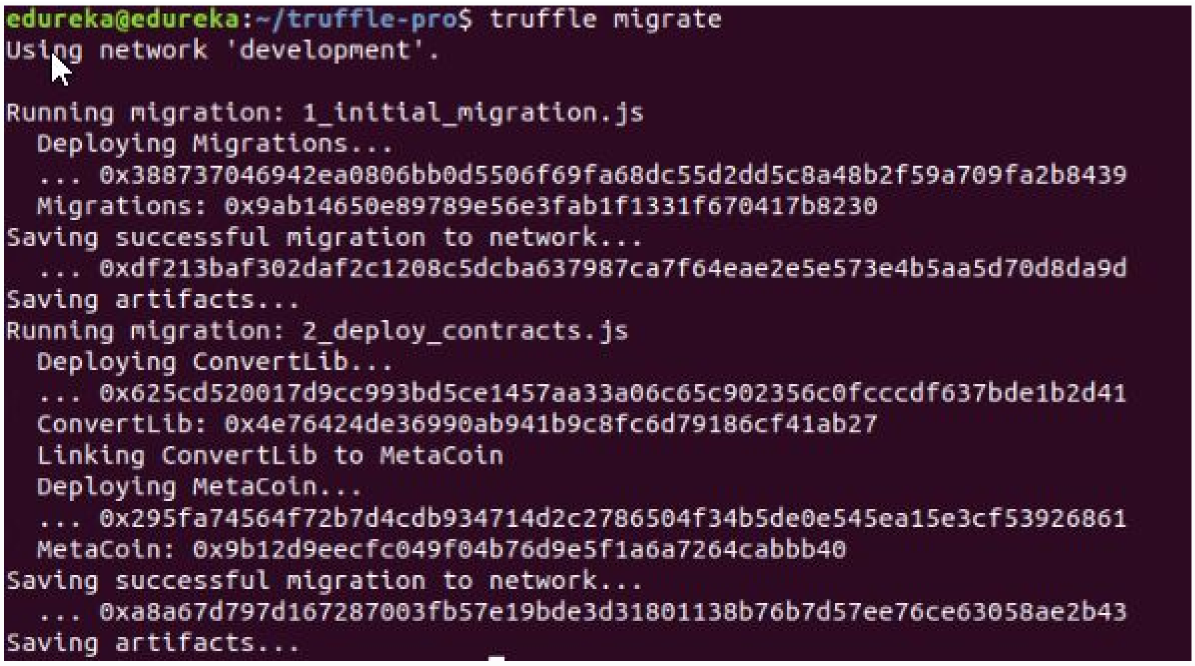
Now, we will have to compile the contract and migrate it to the network. The commands to do this is as follows:

$ truffle compile
$ truffle migrate

You can see that the code was successfully migrated and deployed on the network.



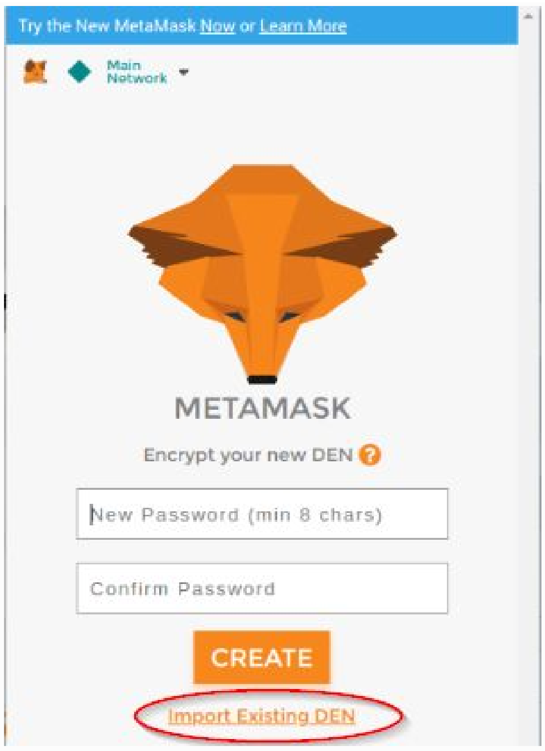
```
edureka@edureka:~/truffle-pro$ truffle compile
Compiling ./contracts/ConvertLib.sol...
Compiling ./contracts/MetaCoin.sol...
Compiling ./contracts/Migrations.sol...
```

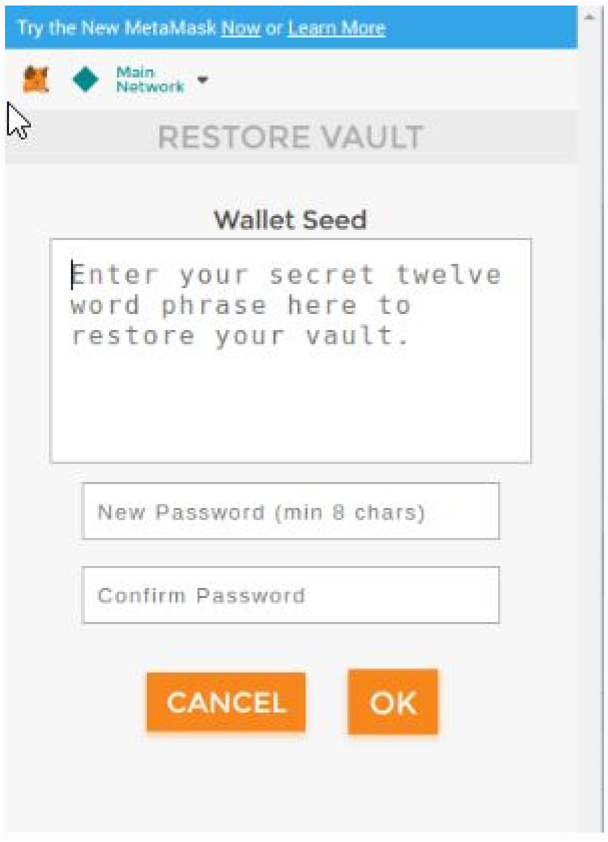


```
edureka@edureka:~/truffle-pro$ truffle migrate
Using network 'development'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0x388737046942ea0806bb0d5506f69fa68dc55d2dd5c8a48b2f59a709fa2b8439
  Migrations: 0x9ab14650e89789e56e3fab1f1331f670417b8230
Saving successful migration to network...
  ... 0xdf213baf302daf2c1208c5dcba637987ca7f64eae2e5e573e4b5aa5d70d8da9d
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying ConvertLib...
  ... 0x625cd520017d9cc993bd5ce1457aa33a06c65c902356c0fcccdf637bde1b2d41
  ConvertLib: 0x4e76424de36990ab941b9c8fc6d79186cf41ab27
  Linking ConvertLib to MetaCoin
  Deploying MetaCoin...
  ... 0x295fa74564f72b7d4cdb934714d2c2786504f34b5de0e545ea15e3cf53926861
  MetaCoin: 0x9b12d9eecfc049f04b76d9e5f1a6a7264cabbb40
Saving successful migration to network...
  ... 0xa8a67d797d167287003fb57e19bde3d31801138b76b7d57ee76ce63058ae2b43
Saving artifacts...
```

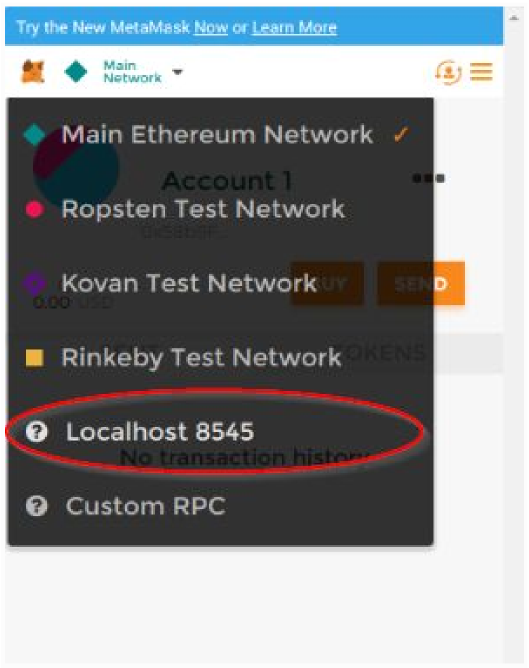
Figure6.4 Migration Successful

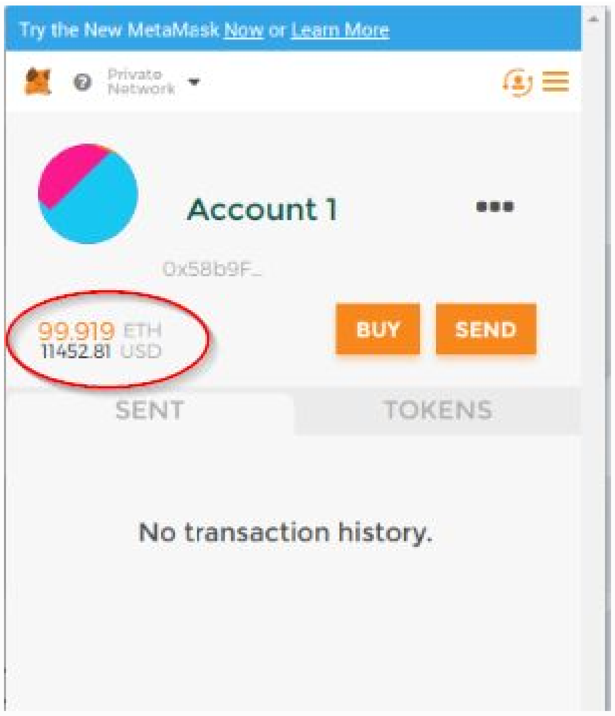
Now, open Chrome browser and click on the MetaMask icon. Click on “Import Existing DEN“. Enter the mnemonic phrase displayed when you executed the **“testrpc”** command, enter the password and click “Ok“.




Try the New MetaMask Now or Learn More
Main Network
RESTORE VAULT
Wallet Seed
Enter your secret twelve word phrase here to restore your vault.
New Password (min 8 chars)
Confirm Password
CANCEL
OK

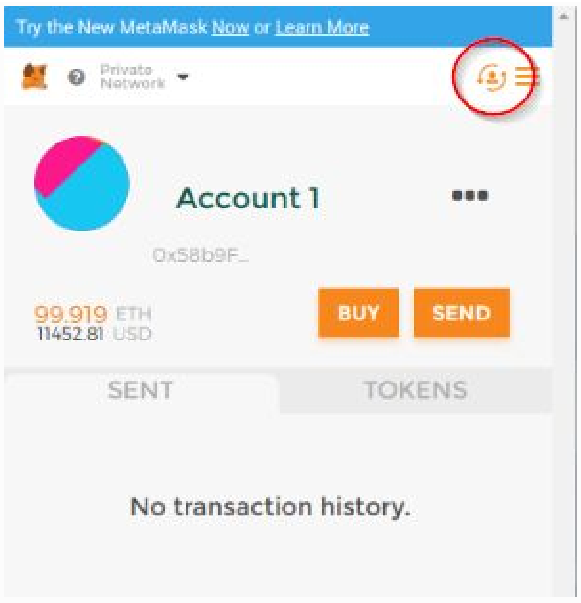
By default, MetaMask runs on the main network. We don't want to spend money just for a demo, right? For that reason, we have to change the network to a private network. In our case, this network is **Localhost 8545.**

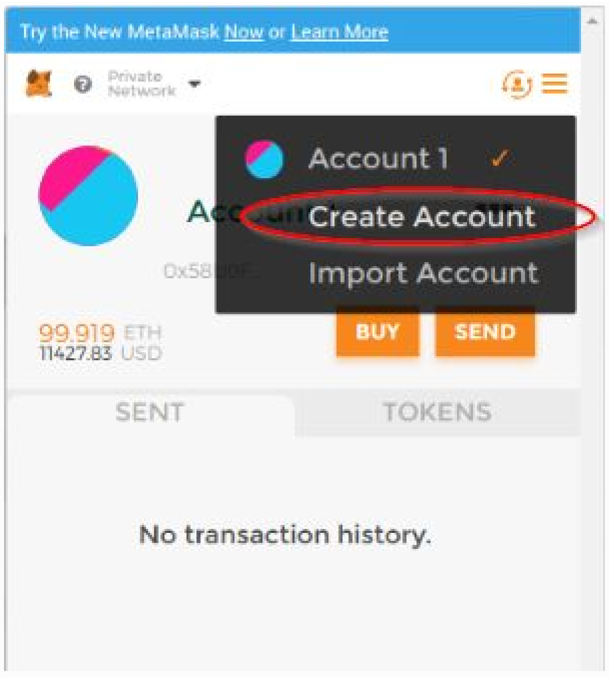




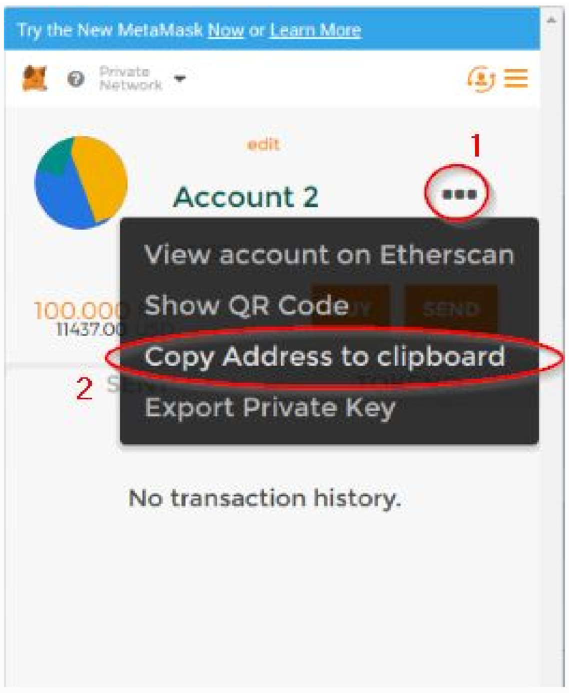
We can now see an account with 99+ ethers in it. "WOW! Free ethers!" Well, to disappoint you, these are not real ethers. These are test ethers provided only for testing purposes and has no real-world value.

We need two accounts to make a transaction: a sender and a receiver. So, let's create a new account. To do this, in the MetaMask plugin, click on "Switch accounts" and then click "Create Account". Your new account is created.

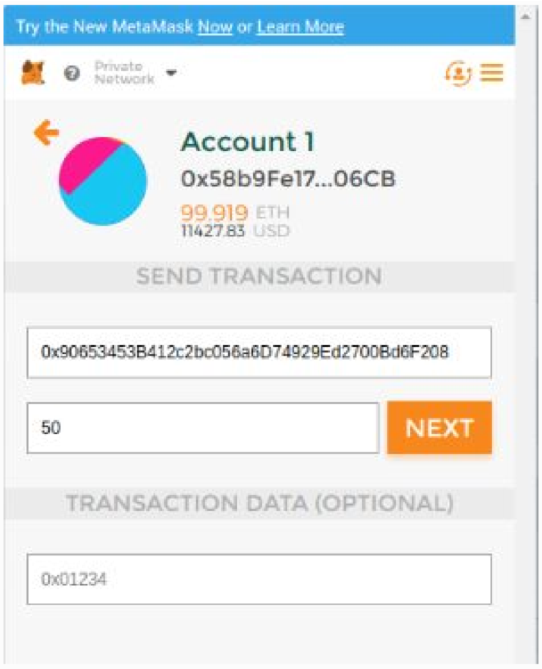




Now, to send ethers to this account, we need to copy the address of this account.

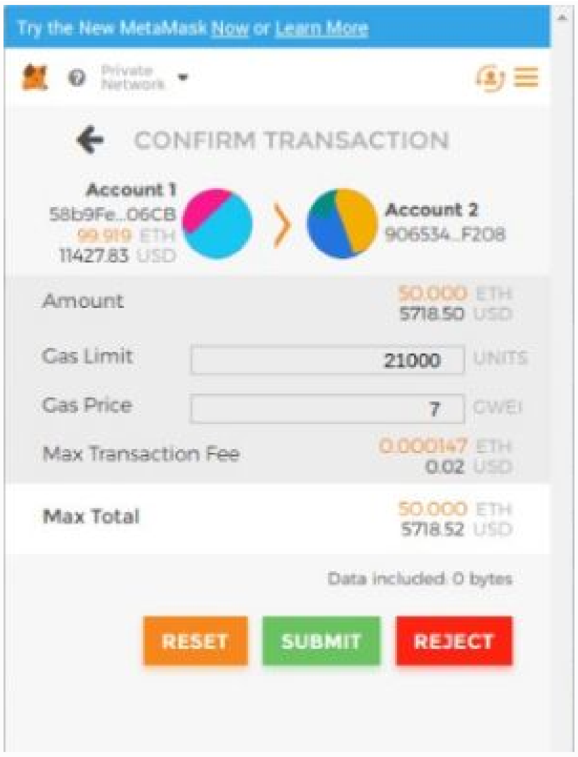

Try the New MetaMask Now or Learn More
Private Network
edit
1
Account 2
View account on Etherscan
Show QR Code
Copy Address to clipboard
2
Export Private Key
No transaction history.

For this Truffle Ethereum tutorial, we will send ethers from Account 1 to Account 2. So, let us switch the account back to Account 1. Here, click on “SEND“, enter the address to which you want to send the account (the address of Account 2 that I copied) and the number of ethers to be sent and click “NEXT“.



Account 1
0x58b9Fe17...06CB
99.919 ETH
11427.83 USD
SEND TRANSACTION
0x90653453B412c2bc056a6D74929Ed2700Bd6F208
50
NEXT
TRANSACTION DATA (OPTIONAL)
0x01234

It will show you a summary of the transaction and ask for confirmation. Click "SUBMIT" and the transaction is done.



CONFIRM TRANSACTION

Account 1
58b9Fe...06CB
99.919 ETH
11427.83 USD

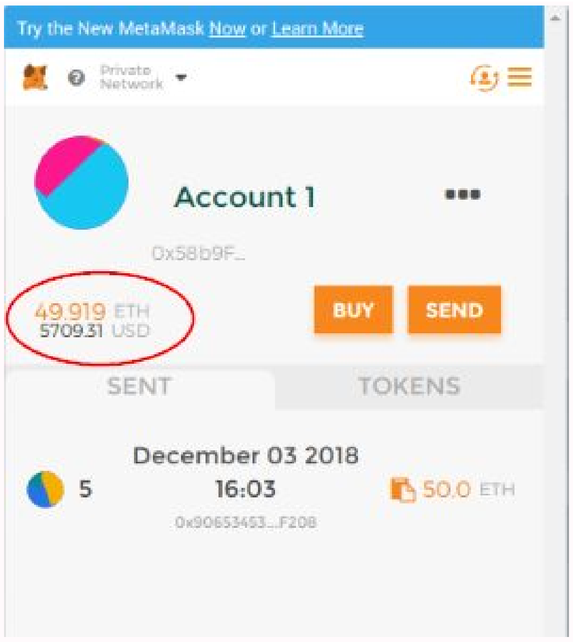
Account 2
906534...F208

Amount		50.000 ETH 5718.50 USD
Gas Limit	21000	UNITS
Gas Price	7	GWEI
Max Transaction Fee		0.000147 ETH 0.02 USD
Max Total		50.000 ETH 5718.52 USD

Data included: 0 bytes

RESET SUBMIT REJECT

We can see now that there are 50 ethers less in Account 1.



To verify the transaction, switch to Account 2. Here, there are 50 ethers more. This shows that 50 ethers were transferred from Account 1 to Account 2.

CONCLUSION

Blockchain has shown its potential for transforming traditional industry with its key characteristics: decentralization,persistency, anonymity and auditability. In this paper, wepresent a comprehensive overview on blockchain. We first give an overview of blockchain technologies including blockchain architecture and key characteristics of blockchain. We then discuss the typical consensus algorithms used in blockchain. We analyzed and compared these protocols in different respects. Furthermore, we listed some challenges and problems that would hinder blockchain development and summarized some existing approaches for solving these problems. Some possible future directions are also proposed. Nowadays blockchain- based applications are springing up and we plan to conduct in-depth investigations on blockchain-based applications in the future.

The training course was very educational and informative. The course instructor Mrs. Beena Ramamurthy taught all the concepts very well. Anyone who takes this course on Smart Contracts and Decentralized Applications can kick start his career in the same.