ABSTRACT

Building an electronic voting system that satisfies the legal requirements of legislators has been a challenge for a long time. Distributed ledger technologies is an exciting technological advancement in the information technology world.

Blockchain technologies offer an infinite range of applications benefiting from sharing economies. This report aims to evaluate the application of blockchain as service to implement distributed electronic voting systems. The core value of a blockchain is that it enables a database to be directly shared without a central administrator. Rather than having some centralized application logic, blockchain transactions have their own proof of validity and authorization to enforce the constraints.

Hence, with the blockchain acting as a consensus mechanism to ensure the nodes stay in sync, transactions can be verified and processed independently. But why is **disintermediation** good for us? Because a database is still a tangible thing even though is just bits and bytes. If the contents of a database are stored in the memory and disk of a particular computer system run by a third party even if it is a trusted organization like banks and governments, anyone who somehow got access to that system can easily corrupt the data within. The report starts by evaluating some of the popular blockchain frameworks that offer blockchain as a service. We then propose an electronic voting system based on blockchain that addresses all limitations we discovered. We are hereby implementing a blockchain-based application which improves the security and decreases the cost of hosting a nationwide election.

CHAPTER 1: INTRODUCTION

1.1 DESCRIPTION - TRADITIONAL VOTING SYSTEM

In every democracy, the security of an election is a matter of national security. The computer security field has for a decade studied the possibilities of electronic voting systems , with the goal of minimizing the cost of having a national election, while fulfilling and increasing the security conditions of an election. From the dawn of democratically electing candidates, the voting system has been based on pen and paper. Replacing the traditional pen and paper scheme with a new election system is critical to limit fraud and having the voting process traceable and verifiable . Electronic voting machines have been viewed as flawed, by the security community, primarily based on physical security concerns. Anyone with physical access to such machine can manipulate the machine, thereby affecting all votes cast on the aforementioned machine.

1.2 ENTER BLOCKCHAIN TECHNOLOGY

A blockchain is a distributed, immutable, incontrovertible, public ledger.
This new technology works through four main features:

(i) The ledger exists in many different locations: No single point of failure in the maintenance of the distributed ledger.

(ii) There is distributed control over who can append new transactions to the ledger.

(iii) Any proposed "new block" to the ledger must reference the previous version of the ledger, creating an immutable chain from where the blockchain gets its name, and thus preventing tampering with the integrity of previous entries.

(iv) A majority of the network nodes must reach a consensus before a proposed new block of entries becomes a permanent part of the ledger.

1.3 CRYPTOGRAPHY AND BLOCKCHAIN

These technological features operate through advanced cryptography, providing a security level equal and/or greater than any previously known database. A lot of people use cryptography on a daily basis without realising it as many popular messaging apps use encryption. It is also one of the core aspects of blockchain technology.

Cryptography is the method of disguising and revealing, otherwise known as encrypting and decrypting, information through complex mathematics. This means that the information can only be viewed by the intended recipients and nobody else. The method involves taking unencrypted data, such as a piece of text, and encrypting it using a mathematical algorithm, known as a cipher. This produces a ciphertext, a piece of information that is completely useless and nonsensical until it is decrypted. This method of encryption is known as symmetric-key cryptography.

An early example of cryptography was the Caesar cipher, used by Julius Caesar to protect Roman military secrets. Each letter in a messages was substituted with the letter 3 spaces to the left in the alphabet, this knowledge was essentially the key that encrypted the message. Caesar's generals knew that to decode the letters they only had to shift each to the right by three, whilst the information remained safe if intercepted by Caesar's enemies. Modern cryptography works on the same level, albeit with far greater levels of complexity.

In blockchain, cryptography is primarily used for two purposes:

1. Securing the identity of the sender of transactions.
2. Ensuring the past records cannot be tampered with.

1.3.1 PUBLIC KEY CRYPTOGRAPHY

Public-key cryptography, also known as asymmetric cryptography, represents an improvement on standard symmetric-key cryptography as it allows information to be transferred through a public key that can be shared with anyone.Rather than using a single key for encryption and decryption, as is the case with symmetric key cryptography, separate keys (a public key and a private key) are used. A combination of a users public key and private encrypt the information, whereas the recipients private key and sender's public key decrypt it. It is impossible to work out what the private key is based on the public key. Therefore, a user can send their public key to anyone without worrying that someone will gain access to their private key. The sender can encrypt files that they can be sure will only be decrypted by the intended party.

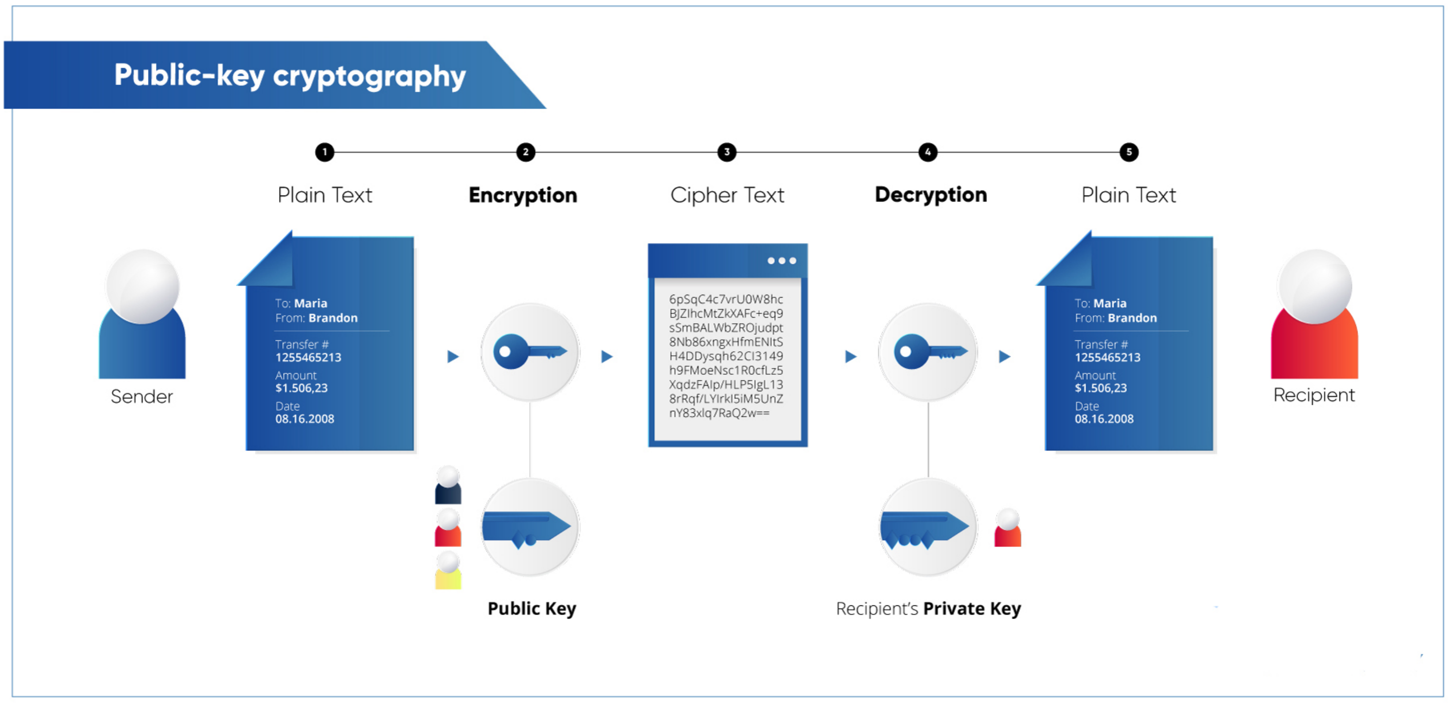


Figure 1.1 Public Key Cryptography

1.4 DIGITAL SIGNATURES

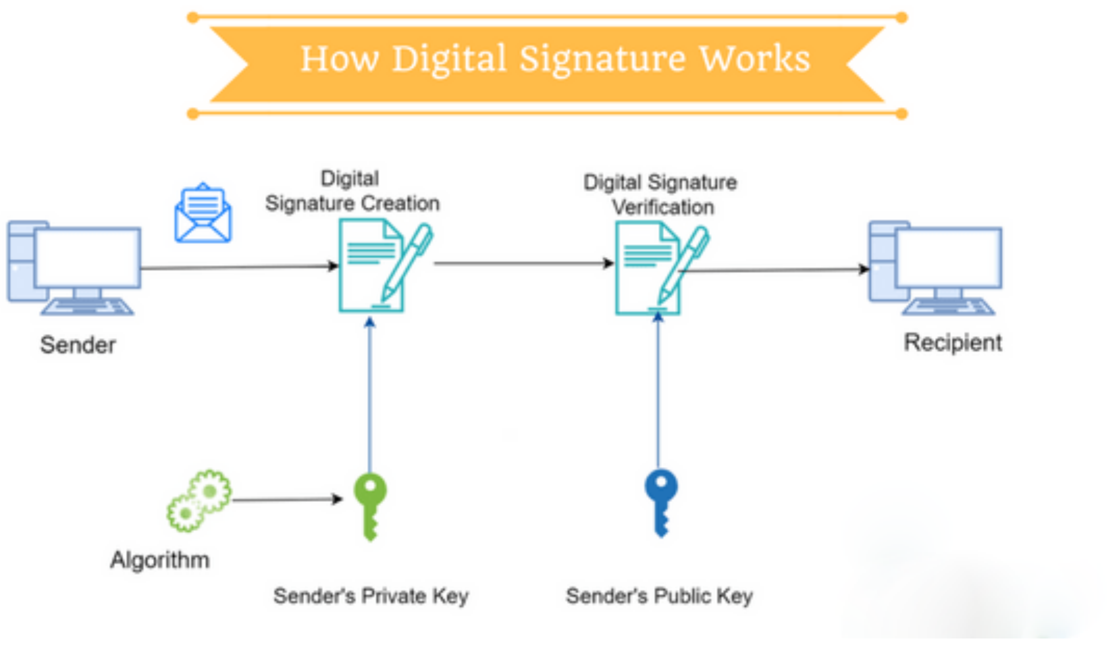

How Digital Signature Works
Digital Signature Creation
Digital Signature Verification
Sender
Recipient
Algorithm
Sender's Private Key
Sender's Public Key

FIGURE 1.2 How a digital signature works

In some ways, digital signatures do what their names suggest: they provide validation and authentication in the same way signatures do, in digital form. In this segment we will discuss how they work as well as how multisignatures (multisigns) can be used to add an extra layer of security.

Digital signatures are one of the main aspects of ensuring the security and integrity of the data that is recorded onto a blockchain. They are a standard part of most blockchain protocols, mainly used for securing transactions and blocks of transactions, transfers of sensitive information, software distribution, contract management and any other cases where detecting and preventing any external tampering is important. Digital signatures utilize asymmetric cryptography, meaning that information can be shared with anyone, through the use of a public key.

1.4.1 DIGITAL SIGNATURES ADVANTAGES

Digital signatures provide three key advantages of storing and transferring information on a blockchain. First of all, they guarantee integrity. Theoretically, encrypted data that is being sent can be altered without being seen by a hacker.

However, if this does happen the signature would change too, thus becoming invalid. Therefore digitally signed data is not only safe from being seen but will also reveal if it has been tampered with, cementing its incorruptibility.
Digital signatures not only secure data but also the identity of the individual sending it. Ownership of a digital signature is always bound to a certain user and as such, one can be sure that they are communicating with whom they intend to.
For example, even the most proficient hacker could not fake another's digital signature as a means of convincing someone else to send money, it is simply mathematically not within the realms of possibility. Therefore digital signatures not only guarantee the data that is being communicated, but also the identity of the individual communicating it.Finally, the fact that private keys are linked to individual users gives digital signatures a quality of non-repudiation. This means that if something is digitally signed by a user, it can be legally binding and entirely associated with that individual. As indicated earlier, this is heavily dependent on there being no doubt that the private key that signed the data was not compromised, used or seen by anyone other than its owner.

1.5 BLOCKCHAIN NODES

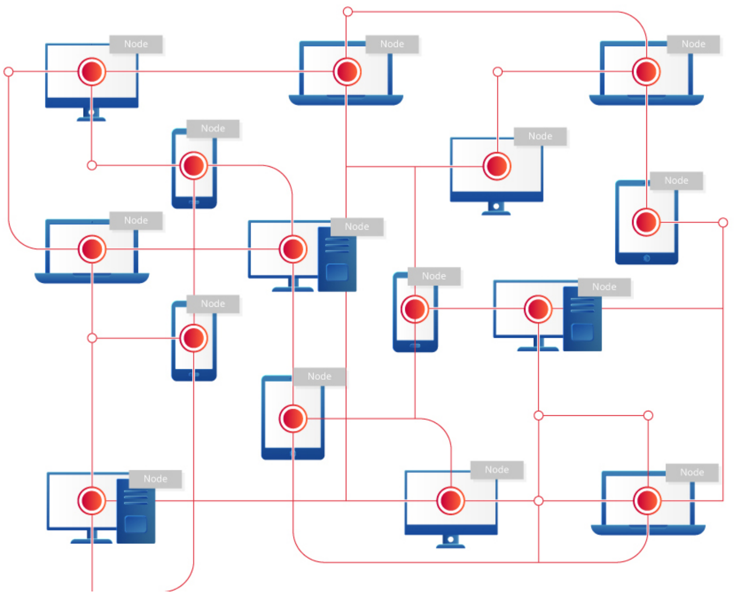


Figure 1.3 Blockchain Nodes Representation

A node is a device on a blockchain network, that is in essence the foundation of the technology, allowing it to function and survive. Nodes are distributed across a widespread network and carry out a variety of tasks.

In this segment of the Academy we will examine the qualities of a node on a blockchain network. A node can be any active electronic device, including a computer, phone or even a printer, as long as it is connected to the internet and as such has an IP address. The role of a node is to support the network by maintaining a copy of a blockchain and, in some cases, to process transactions. Nodes are often arranged in the structure of trees, known as binary trees. Each cryptocurrency has its own nodes, maintaining the transaction records of that particular token.Nodes are the individual parts of the larger data structure that is a blockchain. As the owners of nodes willingly contribute their computing resources to store and validate transactions they have the chance to collect the transaction fees and earn a reward in the underlying cryptocurrency for doing so. This is known as mining.Processing these transactions can require large amounts of computing and processing power, meaning that the average computer's capabilities are inadequate.

1.6 HASHING

The reliability and integrity of blockchain is rooted in there being no chance of any fraudulent data or transactions, such as a double spend, being accepted or recorded. A cornerstone of the technology as a whole and the key components in maintaining this reliability is hashing.

Hashing is the process of taking an input of any length and turning it into a cryptographic fixed output through a mathematical algorithm (Bitcoin uses SHA-256, for example). Examples of such inputs can include a short piece of information such as a message or a huge cache of varying pieces of information such as a block of transactions or even all of the information contained on the internet

1.6.1 SECURING DATA WITH HASHING

Hashing drastically increases the security of the data. Anyone who may be trying to decrypt the data by looking at the hash will not be able to work out the length of the encrypted information based on the hash. A cryptographic hash function needs to have several crucial qualities to be considered useful, these include:

Impossible to produce the same hash value for differing inputs:

This is important because if it were not the case it would be impossible to keep track of the authenticity of inputs.

The same message will always produce the same hash value:
The importance of this is similar to the prior point.
Quick to produce a hash for any given message:
The system would not be efficient or provide value otherwise.**Impossible to determine input based on hash value:**
This is one of the foremost aspects and qualities of hashing and securing data.
Even the slightest change to an input completely alters the hash:
This is also a matter of a security. If a slight change only made a slight difference it would be considerably easier to work out what the input was. The better and more complex the hashing algorithm, the larger the impact of changing an input will be on what the output is.

1.6.2 AN EXPLANATION OF DATA STRUCTURES

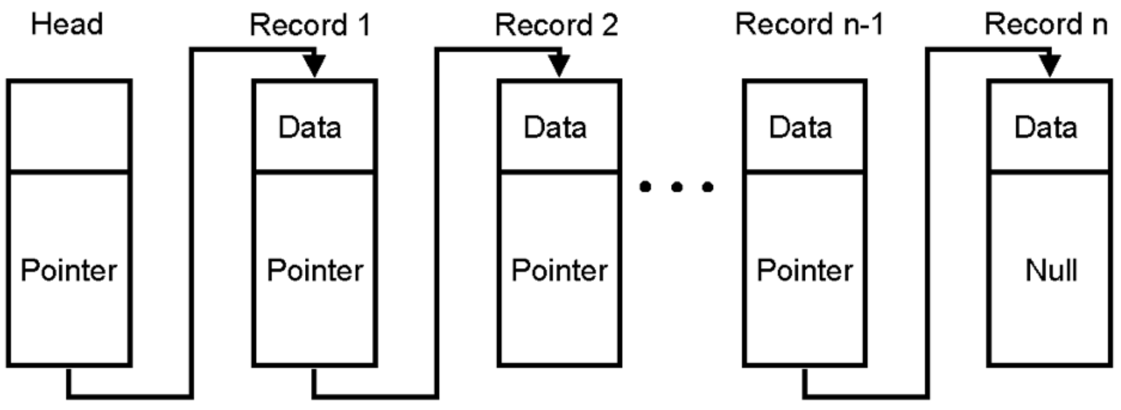


Figure 1.4 Pointer-Node Structure of Linked Blockchain

DataStructures are a specialized way of storing data. The two foremost hashing objects carrying out this function are pointers and linked lists. Pointers store addresses as variables and as such point to the locations of other variables. Linked lists are a sequence of blocks connected to one another through pointers. As such, the variable in each pointer is the address of the next node, with the last node having no pointer and the pointer in the first block, the genesis block, actually lying outside of the block itself. At its simplest, a blockchain is simply a linked list of recorded transactions pointing back to one another through hash pointers.

Hash pointers are where blockchain sets itself apart in terms of certainty as pointers not only contain the address of the previous block, but also the hash data of that block too. As described earlier, this is the foundation of the securenature of blockchain. For example, if a hacker wanted to attack the ninth block in a chain and change its data, he would have to alter the data in all following blocks, as their hash would also change. In essence, this makes it impossible to alter any data that is recorded on a blockchain.

1.6.3 BLOCKCHAIN HASHED STRUCTURE

In blockchain, hashes are used to represent the current state of the world, or to be more precise, the state of a blockchain. As such, the input represents everything that has happened on a blockchain, so every single transaction up to that point, combined with the new data that is being added. What this means is that the output is based on, and therefore shaped by, all previous transactions that have occurred on a blockchain.

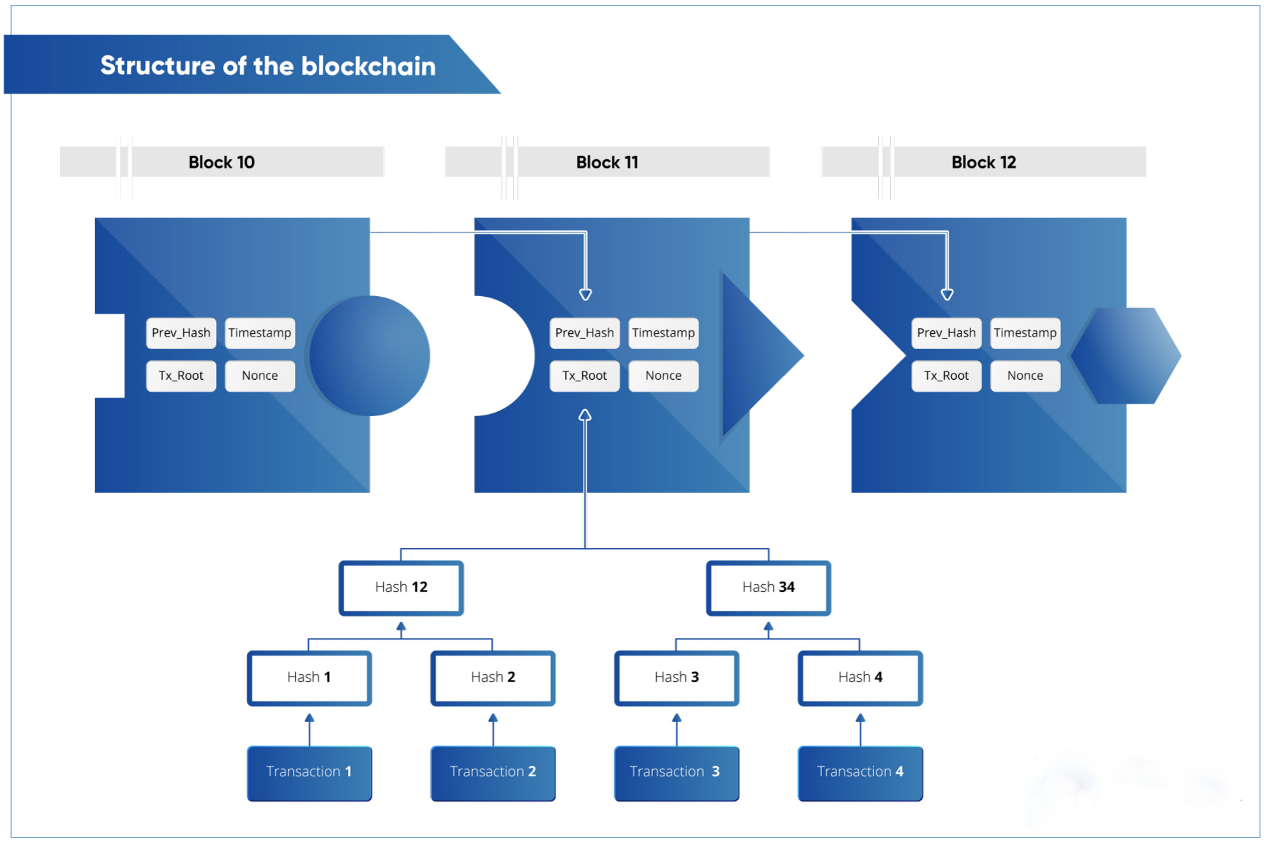


Figure 1.5 Structure of A Blockchain

As mentioned, the slightest change to any part of the input results in a huge change to the output; in this lies the irrefutable security of blockchain technology. Changing any record that has previously happened on a blockchain would change all the hashes, making them false and obsolete. This becomes impossible when the transparentnature of blockchain is taken into account, as these changes would need to be done in plain sight of the whole network.

As such, the input represents everything that has happened on a blockchain, so every single transaction up to that point, combined with the new data that is being added. What this means is that the output is based on, and therefore shaped by, all previous transactions that have occurred on a blockchain.

The first block of a blockchain, known as a genesis block, contains its transactions that, when combined and validated, produce a unique hash. This hash and all the new transactions that are being processed are then used as input to create a brand new hash that is used in the next block in the chain. This means that each block links back to its previous block through its hash, forming a chain back to the genesis block, hence the name blockchain. In this way, transactions can be added securely as long as the nodes on the network are in consensus on what the hash should be.

CHAPTER 2: ETHEREUM

2.1 WHAT IS ETHEREUM ?

Ethereum is a decentralized platform that runs smart contracts: Applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference.These apps run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property. This enables developers to create markets, store registries of debts or promises, move funds in accordance with instructions given long in the past (like a will or a futures contract) and many other things that have not been invented yet, all without a middleman or counterparty risk. The project was bootstrapped via an ether presale in August 2014 by fans all around the world. It is developed by the Ethereum Foundation, a Swiss non-profit, with contributions from great minds across the globe.The intent of Ethereum is to merge together and improve upon the concepts of scripting, altcoins and on-chain metaprotocols, and allow developers to create arbitrary consensus-based applications that have the scalability, standardization, feature-completeness, ease of development and interoperability offered by these WTF different paradigms all at the same time. Ethereum does this by building what is essentially the ultimate abstract foundational layer: a blockchain with a built-in Turing-complete programming language, allowing anyone to write smart contracts and decentralized applications where they can create their own arbitrary rules for ownership, transaction formats and state transition functions.

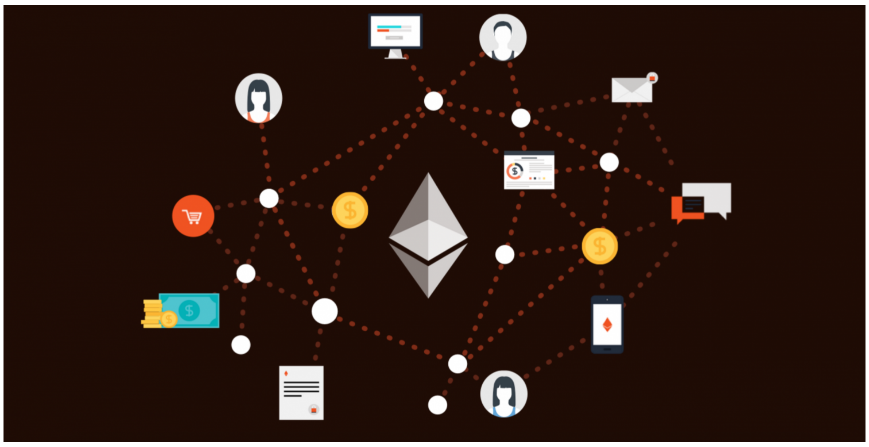


Figure 2.1 High Level View Of Dapp

2.2 SMART CONTRACT

A **smart contract** is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a **contract**. **Smart contracts** allow the performance of credible transactions without third parties. These transactions are trackable and irreversible.

Smart contract is just a phrase used to describe a computer code that can facilitate the exchange of money, content, property, shares, or anything of value. When running on the blockchain a smart contract becomes like a self-operating computer program that automatically executes when specific conditions are met. Because smart contracts run on the blockchain, they run exactly as programmed without any possibility of **censorship, downtime, fraud or third-party interference.**

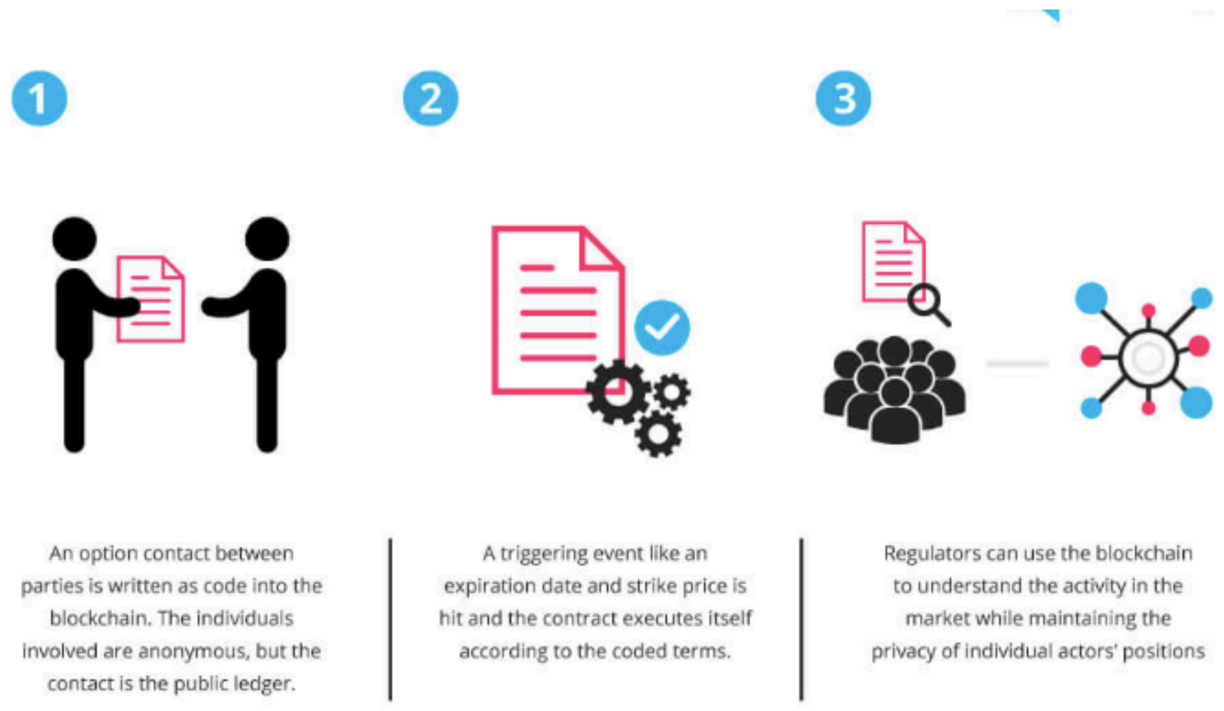

1
2
3
An option contact between parties is written as code into the blockchain. The individuals involved are anonymous, but the contact is the public ledger.
A triggering event like an expiration date and strike price is hit and the contract executes itself according to the coded terms.
Regulators can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

Figure 2.2 Smart Contracts High Level Representation

While all blockchains have the ability to process code, most are severely limited. Ethereum is different. Rather than giving a set of limited operations, Ethereum allows developers to create whatever operations they want. This means developers can build thousands of different applications that go way beyond anything we have seen before.

2.3 ETHEREUM - THE VIRUAL MACHINE

Before the creation of Ethereum, blockchain applications were designed to do a very limited set of operations. Bitcoin and other cryptocurrencies, for example, were developed exclusively to operate as peer-to-peer digital currencies. Developers faced a problem. Either expand the set of functions offered by Bitcoin and other types of applications, which is very complicated and time-consuming, or develop a new blockchain application and an entirely new platform as well. Recognizing this predicament, Ethereum's creator, Vitalik Buterin developed a new approach.

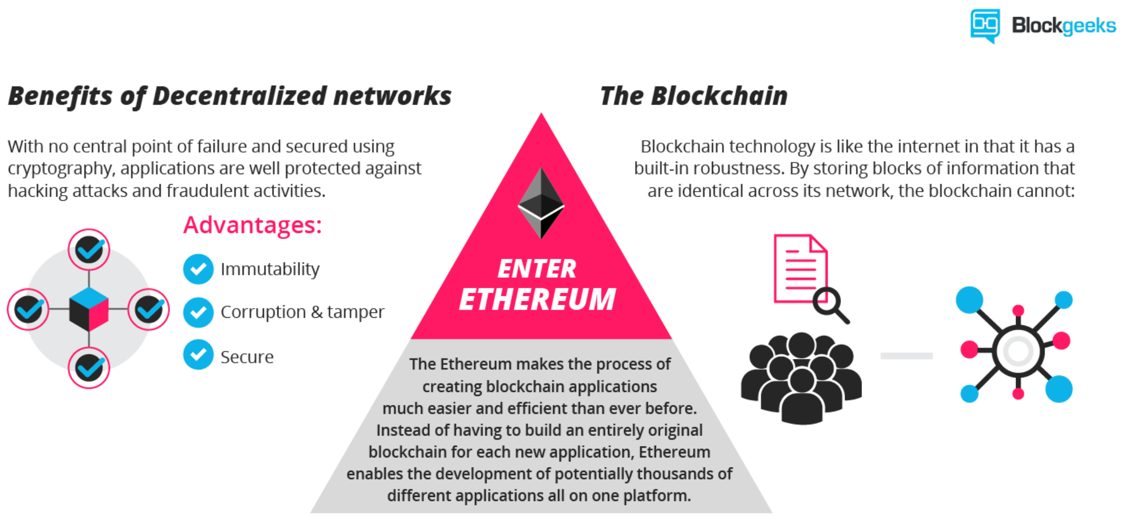

Blockgeeks
Benefits of Decentralized networks
With no central point of failure and secured using cryptography, applications are well protected against hacking attacks and fraudulent activities.
Advantages:
Immutability
Corruption & tamper
Secure
ENTER ETHEREUM
The Ethereum makes the process of creating blockchain applications much easier and efficient than ever before. Instead of having to build an entirely original blockchain for each new application, Ethereum enables the development of potentially thousands of different applications all on one platform.
The Blockchain
Blockchain technology is like the internet in that it has a built-in robustness. By storing blocks of information that are identical across its network, the blockchain cannot:

FIGURE 2.3 Ethereum and Smart Contracts

Ethereum's core innovation, the Ethereum Virtual Machine (EVM) is a Turing complete software that runs on the Ethereum network. It enables anyone to run any program, regardless of the programming language given enough time and memory. The Ethereum Virtual Machine makes the process of creating blockchain applications much easier and efficient than ever before. Instead of having to build an entirely original blockchain for each new application, Ethereum enables the development of potentially thousands of different applications all on one platform.

Ethereum enables developers to build and deploy decentralized applications. A decentralized application or Dapp serve some particular purpose to its users.

Bitcoin, for example, is a Dapp that provides its users with a peer to peer electronic cash system that enables online Bitcoin payments.Because decentralized applications are made up of code that runs on a blockchain network, they are not controlled by any individual or central entity.
Any services that are centralized can be decentralized using Ethereum.
Think about all the intermediary services that exist across hundreds of different industries. From obvious services like loans provided by banks to intermediary services rarely thought about by most people like title registries, voting systems, regulatory compliance and much more.
Ethereum can also be used to build Decentralized Autonomous Organizations (DAO). A DAO is fully autonomous, decentralized organization with no single leader. DAO's are run by programming code, on a collection of smart contracts written on the Ethereum blockchain. The code is designed to replace the rules and structure of a traditional organization, eliminating the need for people and centralized control. A DAO is owned by everyone who purchases tokens, but instead of each token equating to equity shares & ownership, tokens act as contributions that give people voting rights.

2.4 HOW TO ACCESS ETHEREUM?

There are many ways you can plug into the Ethereum network, one of the easiest ways is to use its native Mist browser. Mist provides a user-friendly interface & digital wallet for users to trade & store Ether as well as write, manage, deploy and use smart contracts. Like web browsers give access and help people navigate the internet, Mist provides a portal into the world of decentralized blockchain applications.
There is also the MetaMask browser extension, which turns Google Chrome into an Ethereum browser. MetaMask allows anyone to easily run or develop decentralized applications from their browser. Although initially built as a Chrome plugin, MetaMask supports Firefox and the Brave Browser as well.

2.4.1 web3.js

web3.js is a collection of libraries which allow you to interact with a local or remote ethereum node, using a HTTP connection. It is a collection of libraries that allow you to perform actions like send Ether from one account to another, read and write data from smart contracts, create smart contracts, and so much more!

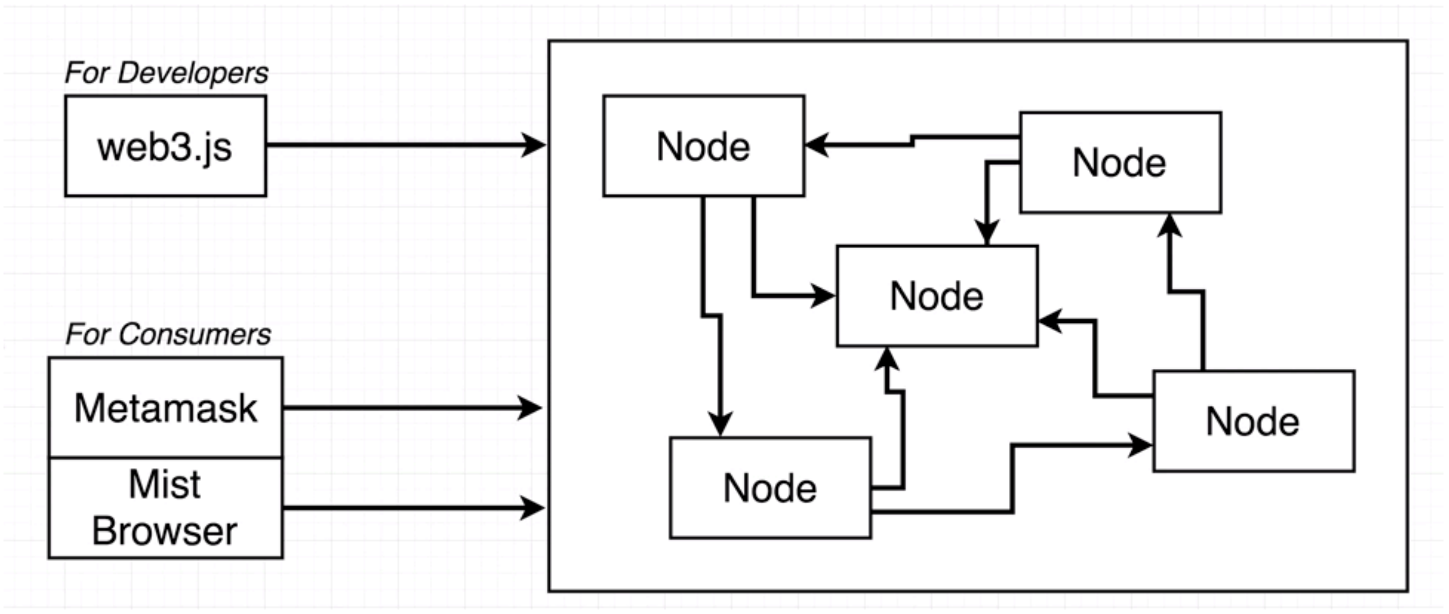


Figure 2.4 How to access an Ethereum Network

The **web3-eth** is for the ethereum blockchain and smart contracts
The **web3-shh** is for the whisper protocol to communicate p2p and broadcast
The **web3-bzz** is for the swarm protocol, the decentralized file storage.
The **web3-utils** contains useful helper functions for Dapp developers

2.4.2 Adding web3.js

First you need to get web3.js into your project. After that you need to create a web3 instance and set a provider. Ethereum supported Browsers like Mist or MetaMask will have a ethereumProvider or web3.currentProvider available.
For web3.js, check Web3.givenProvider. If this property is null you should connect to a remote/local node.

```
var web3 = new Web3(Web3.givenProvider || “ws://localhost:8546");
```

That’s it! now you can use the web3 object.

CHAPTER 3:BLOCKCHAIN AS A SERVICE FOR E-VOTING

3.1 WRITING A SMART CONTRACT

Our contract will include:

- **State Variables**—variables that hold values that are permanently stored on the Blockchain. We will use state variables to hold a list and number of Voters and Candidates.

- **Functions**—Functions are the executables of smart contracts. They are what we will call to interact with the Blockchain, and they have different levels of visibility, internally and externally. Keep in mind that whenever you want to change the value/state of a variable, a transaction must occur—costing Ether. You can also make calls to the Blockchain, which won't cost any Ether because the changes you made will be destroyed.

- **Events**—Whenever an event is called, the value passed into the event will be logged in the transaction's log. This allows Javascript callback functions or resolved promises to view the certain value you wanted to pass back after a transaction. This is because every time you make a transaction, a transaction log will be returned. We will use an event to log the ID of the newly created Candidate, which we'll display (check the first bullet point of Section 3).

- **Struct Types**—This is very similar to a C struct. Structs allow you to hold multiple variables, and are awesome for things with multiple attributes. Candidates will only have their name and party, but you can definitely add more attributes to them.

- **Mappings**—Think of these like hash-maps or dictionaries, where it has a key-value pair. We will use two mappings.

There are a couple more types that aren't listed here, but some of them are a little more complicated. These five encompass many of the structures a smart contract will generally use.

```
struct Voter
 {
bytes32 uid; // bytes32 type are basically strings uint candidateIDVote;
}
```

Describes a Candidate :-

```
struct Candidate
 {
bytes32 name;
bytes32 party;
 }
```

Mapping :-

```
mapping (uint => Candidate) candidates;
mapping (uint => Voter) voters;
```

3.2 INSTANTIATE WEB3.js AND CONTRACTS

With our Smart Contract completed, we now need to run our test blockchain and deploy this contract onto the Blockchain. We'll also need a way to talk to it, which will be via web3.js.

Before we start our test blockchain, we must create a file called contract.js inside the folder `/contracts` that tells it to include your Voting Smart Contract when you migrate.

To start the development Ethereum blockchain, go to your command line and run:

```
truffle develop
```

This will live on your command line. Since Solidity is a compiled language, we must compile it to bytecode first for the EVM to execute.

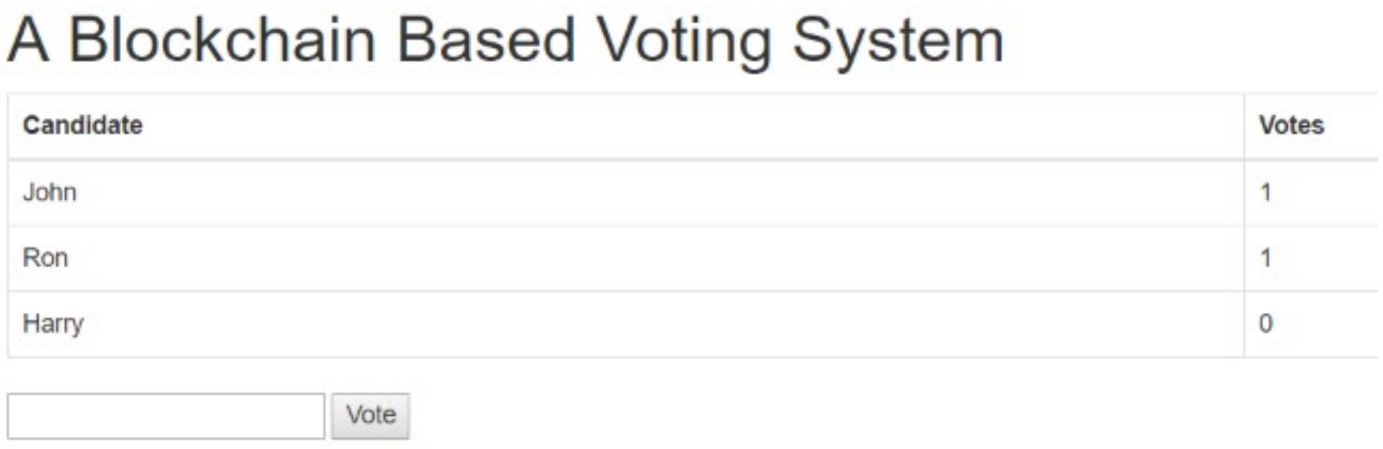
compile

You should see a build/ folder inside your directory now. This folder holds the build artifacts, which are critical to the inner workings of Truffle, so don't touch them! Next, we must migrate the contract. Migrations is a Truffle script that helps you alter the state of your application's contract as you develop. Remember that your contract is deployed to a certain address on the Blockchain, so whenever you make changes, your contract will be located at a different address. Migrations help you do this and also help you move data around.

migrate. Smart contract is now on the Blockchain forever.

3.3 ADDING FUNCTIONALITY

The last thing we'll need to do is to write the interface for the application. This involves the essentials for any web application—HTML, CSS, and Javascript (We've already written a little of the Javascript with creating a web3 instance). First, let's create our HTML file.This is a very simple page, with an input form for user ID, and buttons for Voting and Counting votes. When those buttons are clicked, they will call specific functions that vote, and will find the number of votes for the candidates.There are three important div elements though, with ids: candidate-box, msg, and vote-box, which will hold checkboxes for each candidate, a message, and the number of votes.



A Blockchain Based Voting System

Candidate	Votes
John	1
Ron	1
Harry	0

Figure 3.1 Web View of Casting of Votes

We also import JQuery, Bootstrap, and app.js Now, we just need to interact with the Contract and implement the functions for voting and counting the number of votes for each candidate. JQuery will manipulate the DOM.

REFERENCES

[1] “Bitcoin: A Peer-to-Peer Electronic Cash System” , white paper of bitcoin , Satoshi Nakamoto, 2009

[2] “A Next-Generation Smart Contract and Decentralized Application Platform” An introductory paper to Ethereum, introduced before launch.

[3] “Blockchain-Based E-Voting System” Friðrik Þ. Hjálmarsson, Gunnlaugur K. Hreiðarsson, School of Computer Science Reykjavik University, Iceland

[4] “DECENTRALIZED AUTONOMOUS ORGANIZATION TO AUTOMATE GOVERNANCE” , ,CHRISTOPH JENTZSCH

[5] Web3 javascript documentation

[6] Ethereum Documentation

[7] “Blockchain Architecture Design and Use Cases”, Prof. Sandip Chakraborty , Dr. Praveen Jayachandran