E-Voting DApp

A Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Bachelor of Technology
in
Information Technology

by

Registration No.	**Name**
20158070	Sahildeep Singh Raina
20158094	Sarvaigari Govardhini
20158014	Ayush Jain
20158098	Gourav Kumar
20158008	Tejaswini Sundar

to the
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY
ALLAHABAD
November, 2018

UNDERTAKING

We declare that the work presented in this report titled "*E-Voting DApp*", submitted to the Computer Science and Engineering Department, Motilal Nehru National Institute of Technology Allahabad, for the award of the ***Bachelor of Technology*** degree in ***Information Technology***, is our original work. We have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, we accept that our degree may be unconditionally withdrawn.

November, 2018
Allahabad

Sahildeep Singh Raina,
Sarvaigari Govardhini,
Ayush Jain,
Gourav Kumar,
Tejaswini Sundar

CERTIFICATE

Certified that the work contained in the report titled "*E-Voting DApp*", by *Sahildeep Singh Raina, Sarvaigari Govardhini, Ayush Jain, Gourav Kumar and Tejaswini Sundar*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Dr. Divya Kumar)
Computer Science and Engineering Dept.
M.N.N.I.T, Allahabad

November, 2018

Preface

The most stable and efficient governments practice democracy, and in such a democratic nation, elections play a pivotal role. As such, the issue with the current ballot system is that it can be manipulated easily. The proposed system looks to eliminate the aspect of trust from an election to make it more transparent and secure. The system uses existing technologies like client-server architecture of web-based applications integrated with a blockchain system to ensure aspects such as transparency, security and efficiency without sacrificing the privacy of voters. The cost of building the system is substantially less than the cost of running a ballot based system, with increased robustness. There are substantial social benefits to using this system as well, such as easier and quicker voting process which will lead to higher voter turnout. This system can be implemented for a larger number of countries as the internet penetration in the world increases. In a not so distant future, systems such as these might dominate elections all over the world.

Acknowledgements

We would like to take this opportunity to express our deep sense of gratitude to all who helped us directly or indirectly for our thesis work. First, we would like to thank our supervisor, Dr. Divya Kumar, for being an ideal mentor everyone wishes for. His advise, encouragement and critiques are a source of innovative ideas, inspiration and the cause behind the successful completion of this dissertation. The confidence shown on us by him was the biggest source of inspiration for us. His perpetual energy and enthusiasm in research has motivated us. In addition, he is always accessible and willing to help his students with their research. As a result, research life became smooth and rewarding for us. It has been a privilege working with him for last six months. He gave us ample opportunities to explore ourselves.

We wish to express our sincere gratitude to Prof. Rajeev Tripathi, Director, MNNIT Allahabad, Allahabad and Prof. A. K. Singh, Head, Computer Science and Engineering Department, for providing us all the facilities required for the completion of this thesis work. We would also like to thank our friends for their constant motivation, advice and support.

Contents

List of Figures

Chapter 1

Introduction

In today's world, widespread mistrust towards the government and interference in countries' electoral processes by external actors have made democratic processes of voting more critical than ever. Democratic countries have been experiencing dictatorial regimes, which have introduced widespread terror among their people. People have had their human rights violated and their fundamental freedom provided by their constitutions taken away. In such an atmosphere, having a fair and transparent election is something that is paramount for the freedom most of us enjoy today.

1.1 Motivation

1.1.1 Wonderful Minds

The first work on a cryptographically secured chain of blocks was described in 1991 by Stuart Haber and W. Scott Stornetta [1]. They wanted to implement a system where document timestamps could not be tampered with. In 1992, Bayer, Haber and Stornetta [2] incorporated Merkle trees to the design, which improved its efficiency by allowing several document certificates to be collected into one block.

The first blockchain [8] was conceptualized by a person (or group of people) known as Satoshi Nakamoto [3] in 2008. Nakamoto improved the design in an important way using a Hashcash-like method to add blocks to the chain without requiring them to be signed by a trusted party. The design was implemented the following year by Nakamoto as a core component of the cryptocurrency bitcoin, where it serves as the public ledger for all transactions on the network.

In August 2014, the bitcoin blockchain file size, containing records of all transactions that have occurred on the network, reached 20 GB (gigabytes). In January 2015, the size had grown to almost 30 GB, and from January 2016 to January 2017, the bitcoin blockchain grew from 50 GB to 100 GB in size [4].

1.1.2 Historical Evidence

The pitfalls of the current system of ballot voting are exploited by people or organizations looking to gain power. In the African countries of Uganda [5] and Kenya [6], there has been widespread controversy over their elections in recent years. The election of 1946 in Romania [7] was heavily rigged. The communists took over Romania and abolished multi-party system to gain complete control of the country. In India, almost all elections are somehow tainted with malpractices.

1.1.3 Need of the Hour

These instances of controversial elections could all have been avoided if the counting process was fair, transparent and verifiable. The current ballot system does offer anonymity to the voter, but the counting process is not transparent. People are supposed to trust the result which is provided by the Election Commission or a government body. This makes the process of counting a major vulnerability in the current process. There are also other major electoral scams such as voter fraud, ballot stuffing and booth capturing. All these make it very difficult for organizers of an election to distinguish between actual votes and fraudulent votes.

All these points, along with other historical evidence, point towards the need of a better voting system.

1.2 What is Blockchain?

A blockchain [8], originally block chain, is a growing list of records, called blocks, which are linked using cryptography. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data (generally represented as a merkle tree root hash).

By design, a blockchain is resistant to modification of the data. It is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way". For use as a distributed ledger, a blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and validating new blocks. Once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks, which requires consensus of the network majority. Although blockchain records are not unalterable, blockchains may be considered secure by design and exemplify a distributed computing system with high Byzantine fault tolerance [10]. Decentralized consensus has therefore been claimed with a blockchain.

Blockchain was invented by Satoshi Nakamoto in 2008 to serve as the public transaction ledger of the cryptocurrency bitcoin. The invention of the blockchain for bitcoin made it the first digital currency to solve the double-spending problem without the need of a trusted authority or central server. The bitcoin design has inspired other applications, and blockchains which are readable by the public are widely used by cryptocurrencies. Blockchain is considered a type of payment rail. Private blockchains have been proposed for various business uses.

1.2.1 Current Work In Finance And Banking

- Deutsche Bank [11]: Exploring the use cases of blockchain in currency settlement, trade processing, derivative contracts etc.

- US Federal Reserve [12]: Developing a digital cash system using blockchain with the help of IBM.

- Barclays Bank [13]: Runs dedicated labs in London for bitcoin and blockchain entrepreneurs, coders and businesses.

- Citigroup [14]: Implementing a distributed ledger technology using blockchain and a test currency called citicoin'.
- NASDAQ [15]: Using Nasdaq Financial Framework, it has opened up its blockchain services to more than 100 of its market operator clients.

1.2.2 Why Blockchain?

Encrypted and distributed database doesn't allow changes to the data (ledger) once it is written unless a consensus is achieved against it. Thus it reduces the possibility of security breaches by even it's administrators. This makes blockchain invaluable for organizations trying to accomplish a secure system.

Below are some of the important benefits associated with implementing blockchain technology:

1. Trustworthy system: Data structures built using blockchain allows users to make and verify transactions without a third-party involvement. This strongly reduces the risk of a backdoor transaction and unauthorized intervention. The more widespread the environments, the more difficult it is to tinker with data. Modification of historical data is only possible if one has a large team working simultaneously across data centers. This greatly reduces the possibility of data tampering and creates a robust system.

2. Transparency: The distributed ledger structure gives the control of all their information and transactions to the users. Blockchain data is complete, accurate and consistent with all the members. Changes to the public blockchain are accessible to all the members, thus creating a transparent system. Moreover, use of a single public ledger to record transactions reduces the clutter and complications of multiple ledgers.

3. Faster transactions: Physical markets even working with digital documentation takes longer time to execute transactions. Interbank transactions, especially in non-working can potentially take days for clearing and final settlement. Blockchain transactions can reduce transaction times to minutes and are processed 24/7. An

instantaneous settlement would transform an industry such as transportation and energy, potentially saving billions from increased back-office efficiency and automation.

4. Reduced transaction costs: A transaction system built using blockchain eliminates third party intermediaries and overhead costs for exchanging assets. According to Euro Banking Association (EBA)'s report on Cryptotechnologies [16], a major IT innovation and catalyst for change, blockchain technology can be leveraged by banks to reduce governance and audit costs and to provide better products with faster time to market.

Blockchain technology is more efficient, transparent and cost-effective for new generation of transactional applications.

Chapter 2

Related Work

Secure e-voting is considered as one of the most difficult problems in security literature as it involves many security requirements. To satisfy these security requirements, cryptographic techniques are mostly applied in constructing a secure e-voting system. In this section, we discuss the existing voting systems based on traditional public bulletin and blockchain technology.

2.1 Public Bulletin Based Voting Systems

In the following, we outline the key cryptographic techniques used in public bulletin based voting systems and how such techniques address the corresponding security requirements

2.1.1 Homomorphic Encryption

Homomorphism [17] feature allows one to operate on ciphertexts without decrypting them. For a voting system, this property allows the encrypted ballots to be counted by any third party without leaking any information in the ballot. Typical cryptosystems applied in a voting system are Paillier encryption and ElGamal encryption.

2.1.2 Mix-net

Mix-net [19] was proposed in 1981 by Chaum. The main idea of mix-net is to perform a re-encryption over a set of ciphertexts and shuffle the order of those ciphertexts. Mix node only knows the node that it immediately received the message from and the immediate destination to send the shuffled messages to. The voting systems proposed in apply mix-net to shuffle the ballots from different voters, thus the authority cannot relate a ballot to a voter. For the mix-net based voting systems, they need enough amount of mix nodes and ballots to be mixed.

2.1.3 Zero-Knowledge Proof

Zero-knowledge proof is often employed in a voting system to let the prover to prove that the statement is indeed what it claimed without revealing any additional knowledge of the statement itself. In a voting system, the voter should convince the authority that his ballot is valid by proving that the ballot includes only one legitimate candidate without revealing the candidate information.

2.1.4 Blind Signature And Linkable Ring Signature

Voting systems like to employ blind signature to convince the tallying centre that the ballot is from a valid voter while not revealing the owner of the ballot. Simultaneously, the authority who signs the ballot learns nothing about the voter's selections. In blind signature, both voters and tallying centre must trust the signer. If the signer is compromised, the signature scheme may stop working. Unlike blind signature, linkable ring signature is proposed to avoid the untrusted signer. Instead, it needs a certain number of voters to participate in the signing process. By comparing the linkability tag, the authority can easily tell whether this voter has already voted. When the voter signs on the ballot, he/she needs to include other voters' public keys to make his/her signature indistinguishable from other voters' signatures.

2.2 Other Attempts

There has been a lot of work on remote e-voting protocols using cryptographic tools. In some cases, a trusted third party (TTP) is involved to make the implementation and control of e-voting systems easier. However, a powerful TTP may also become the vulnerable spot of the whole system. A few efforts have been made to combine an e-voting protocol with the blockchain paradigm to design a voting protocol without a TTP, which provides anonymity and verifiability as well. Zhao and Chan proposed a voting protocol [18] in 2015, which introduces a reward/penalty scheme for correct or incorrect behaviors of voters. Although the protocol has some limitations, this is the first attempt to combine e-voting with blockchain. Later in 2016, Lee, James, Ejeta and Kim proposed an e-voting protocol [20], which involves a TTP into blockchain to preserve voters' choices. Very recently, using Bitcoin, Bistarelli, Mantilacci, Santancini and Santini proposed another e-voting protocol. This protocol divides the organizer of elections into two different parts - the Authentication Server (AS) and the Token Distribution Server (TDS), to protect voters' privacy. However, there remain some problems in this protocol, for example, it is difficult to inspect these two parts' behaviors, and it limits the extension of the voting scheme.

Chapter 3

Proposed Work

The proposed system can be divided into three parts:

- Create Blockchain Backbone
- Create Web Based Application
- Connect The Two

The following concepts and topics are crucial in understanding the mechanics of the e-voting application.

3.1 Blockchain

A blockchain is a digital record of transactions. The name comes from its structure, in which individual records, called blocks, are linked together in a single list, called a chain. Blockchains are used for recording transactions made with cryptocurrencies, such as Bitcoin, and have many other applications.

Each transaction added to a blockchain is validated by multiple computers on the Internet. These systems, which are configured to monitor specific types of blockchain transactions, form a peer-to-peer network. They work together to ensure each transaction is valid before it is added to the blockchain. This decentralized network of computers ensures a single system cannot add invalid blocks to the chain.

When a new block is added to a blockchain, it is linked to the previous block using a cryptographic hash generated from the contents of the previous block. This ensures the chain is never broken and that each block is permanently recorded. It is also intentionally difficult to alter past transactions in blockchain since all the subsequent blocks must be altered first.

3.1.1 Types of blockchains

Currently, there are three types of blockchain networks - public blockchains, private blockchains and consortium blockchains.

▮ *Public blockchains*

A public blockchain has absolutely no access restrictions. Anyone with an internet connection can send transactions[disambiguation needed] to it as well as become a validator (i.e., participate in the execution of a consensus protocol).[self-published source?] Usually, such networks offer economic incentives for those who secure them and utilize some type of a Proof of Stake or Proof of Work algorithm.

Some of the largest, most known public blockchains are Bitcoin and Ethereum.

▮ *Private blockchains*

A private blockchain is permissioned. One cannot join it unless invited by the network administrators. Participant and validator access is restricted.

This type of blockchains can be considered a middle-ground for companies that are interested in the blockchain technology in general but are not comfortable with a level of control offered by public networks. Typically, they seek to incorporate blockchain into their accounting and record-keeping procedures without sacrificing autonomy and running the risk of exposing sensitive data to the public internet.

▮ *Consortium blockchains*

A consortium blockchain is often said to be semi-decentralized. It, too, is permissioned but instead of a single organization controlling it, a number of companies might each operate a node on such a network. The administrators of a consortium chain restricts users' reading rights as they see fit and only allow a limited set of trusted nodes to execute a consensus protocol.

3.2 Smart Contracts [21]

A smart contract [21] 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.

Proponents of smart contracts claim that many kinds of contractual clauses may be made partially or fully self-executing, self-enforcing, or both. The aim of smart contracts is to provide security that is superior to traditional contract law and to reduce other transaction costs associated with contracting. Various cryptocurrencies have implemented types of smart contracts.

Smart contracts were first proposed by Nick Szabo, who coined the term. With the present implementations, based on blockchains, "smart contract" is mostly used more specifically in the sense of general purpose computation that takes place on a blockchain or distributed ledger. In this interpretation, used for example by the Ethereum Foundation [22] or IBM [23], a smart contract is not necessarily related to the classical concept of a contract, but can be any kind of computer program.

We use ethereum based smart contract in this project.

3.3 Web3.0

Web3.0 [24] enables you to fulfill the second responsibility: developing clients that interact with The Ethereum Blockchain. It is a collection of libraries that allow you to perform actions like send Ether from one account to another, read and write data

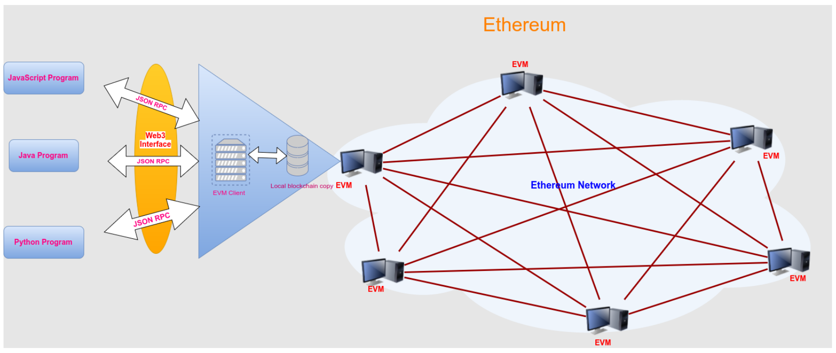


Figure 1: Communication between Client and Ethereum

from smart contracts, create smart contracts, and so much more!

People with web development background might have used jQuery to make Ajax calls to a web server. That's a good starting point for understanding the function of Web3.0. Instead of using a jQuery to read and write data from a web server, you can use Web3.0 to read and write to The Ethereum Blockchain.

The following explains how you can use Web3.0 to talk to The Ethereum Blockchain. Here is a diagram of how a client talks to Ethereum:

Web3.0 talks to The Ethereum Blockchain with JSON RPC, which stands for "Remote Procedure Call" protocol. Ethereum is a peer-to-peer network of nodes that stores a copy of all the data and code on the blockchain. Web3.0 allows us to make requests to an individual Ethereum node with JSON RPC in order to read and write data to the network. It's kind of like using jQuery with a JSON API to read and write data with a web server.

3.4 DApp

3.4.1 Web App

Web applications use a combination of server-side scripts (PHP and ASP) to handle the storage and retrieval of the information, and client-side scripts (JavaScript and HTML) to present information to users. This allows users to interact with the company using online forms, content management systems, shopping carts and more. In

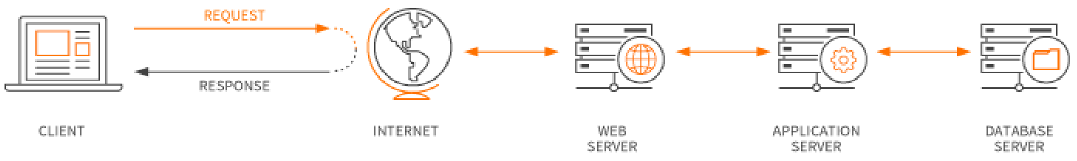


Figure 2: Web Application

addition, the applications allow employees to create documents, share information, collaborate on projects, and work on common documents regardless of location or device.

3.4.2 DApp

Decentralized applications (DApps) are applications that run on a P2P network of computers rather than a single computer. DApps, have existed since the advent of P2P networks. They are a type of software program designed to exist on the Internet in a way that is not controlled by any single entity.

Decentralized applications don't necessarily need to run on top of a blockchain network. BitTorrent [25], Popcorn Time [26], BitMessage [27], Tor [28], are all traditional DApps that run on a P2P network, but not on a Blockchain (which is a specific kind of P2P network). As opposed to simple smart contracts, in the classic sense of Bitcoin, that sends money from A to B, DApps have an unlimited number of participants on all sides of the market.

3.4.3 Difference between DApps & Smart Contracts

DApps are a 'blockchain enabled' website, where the Smart Contract is what allows it to connect to the blockchain. The easiest way to understand this is to understand how traditional websites operate.

The traditional web application uses HTML, CSS and Javascript to render a page. It will also need to grab details from a database utilizing an API. When you go onto Facebook [29], the page will call an API to grab your personal data and display them on the page.

Traditional websites: Front End $\rightarrow API \rightarrow Database$

DApps are similar to a conventional web application. The front end uses the exact same technology to render the page. The one critical difference is that instead of an API connecting to a Database, you have a Smart Contract connecting to a blockchain.

DApp enabled website: Front End $\rightarrow SmartContract \rightarrow Blockchain$

As opposed to traditional, centralized applications, where the back end code is running on centralized servers, DApps have their back end code running on a decentralized P2P network. Decentralized applications consist of the whole package, from back end to front end. The smart contract is only one part of the DApp:

- Front end (what you can see), and
- Back end (the logic in the background)

A smart contract, on the other hand, consists only of the back end, and often only a small part of the whole DApp. That means if you want to create a decentralized application on a smart contract system, you have to combine several smart contracts and rely on 3rd party systems for the front-end.

3.5 Steps Involved

The following steps were involved in creating this e-voting application:

3.5.1 Blockchain Back End

The background logic is written in Solidity [30], a programming language made specifically to write smart contracts that are based on ethereum blockchain.

3.5.2 Web Based Front End

The front of the application, which the user interacts with is written in HTML, CSS and JavaScript. The front end provides required functionalities and works in web browsers.

3.5.3 Bridge Between Front End and Back End

The front end is connected to the back end via the Web3.0 library written in JavaScript. It connects the functions provided by the smart contract to be available to the front end and allows the front end to initiate transactions.

Chapter 4

Experimental Setup and Results Analysis

This project involves creating a front end for the application, creating a blockchain back-support and connecting the two.

4.1 Requirements

For the project to run, some basic conditions must be met.

4.1.1 Hardware Requirements

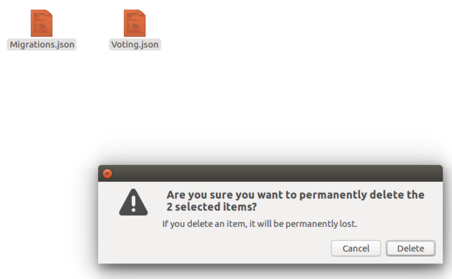
The following hardware requirements must be met:

- A processing unit to serve as server for webpage
- Processing units that can connect to a network

4.1.2 Software Requirements

This project also has some software requirements:

- Web3.0 [24], a JavaScript Library
- Solidity [30], for writing smart contracts



Are you sure you want to permanently delete the 2 selected items?

If you delete an item, it will be permanently lost.

Figure 3: Preprocessing

- NodeJs [31], for deploying webpages
- Truffle [32], for compiling smart contracts and creating artifacts
- MetaMask [33] (optional), for testing the smart contract functionalities
- Operating system that supports these dependencies, like Windows or Linux

4.2 Working

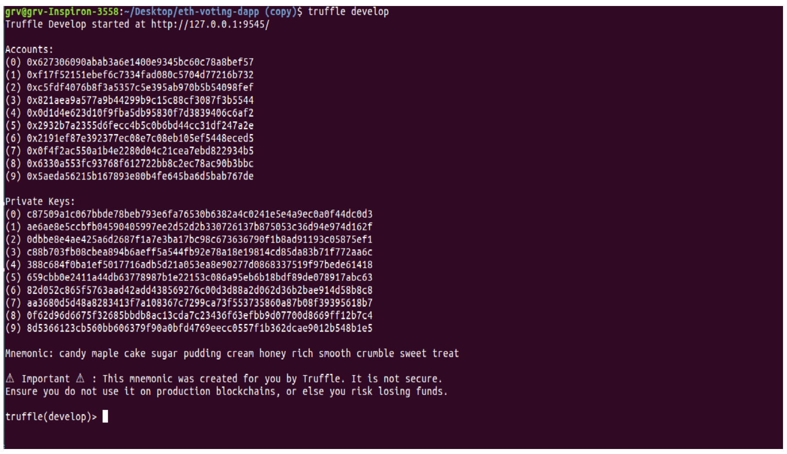
4.2.1 Preprocessing

In this step, previous artifacts are deleted so that new voting instances can be created, as shown in Figure 3.

4.2.2 Start Instance Of Blockchain

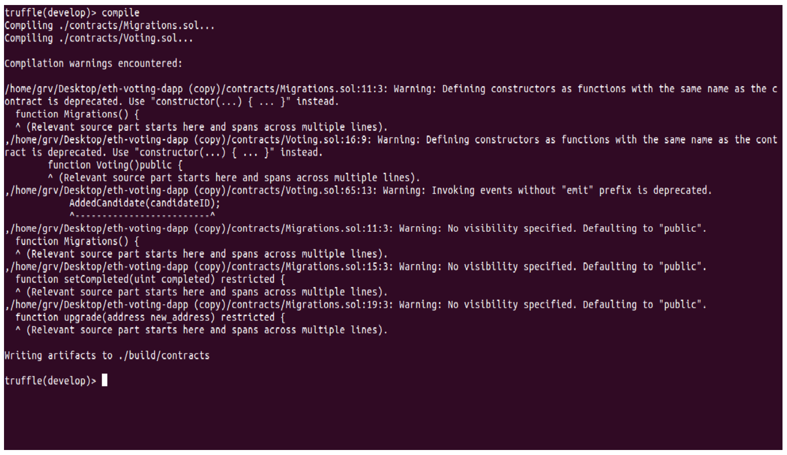
In this step, the instance of blockchain is started. This creates a private blockchain and gives the mnemonics in case it is to be tested in web browsers. We are also provided with 10 accounts in the blockchain with some initial gas. Since this is a private blockchain, money is not involved. As such, this blockchain is used for testing purposes only. The command used, as shown in Figure 4, is:

```
truffle develop
```



```
grv@grv-Inspiron-3558:~/Desktop/eth-voting-dapp (copy)$ truffle develop
Truffle Develop started at http://127.0.0.1:9545/

Accounts:
(0) 0x627306090abab3a6e1400e9345bc60c78a8bef57
(1) 0xf17f52151ebef6c7334fad080c5704d77216b732
(2) 0xc5fdf4076b8f3a5357c5e395ab970b5b54098fef
(3) 0x821aea9a577a9b44299b9c15c88cf3087f3b5544
(4) 0x0d1d4e623d10f9fba5db95830f7d3839406c6af2
(5) 0x2932b7a2355d6fecc4b5c0b6bd44cc31df247a2e
(6) 0x2191ef87e392377ec08e7c08eb105ef5448eced5
(7) 0x0f4f2ac550a1b4e2280d04c21cea7ebd822934b5
(8) 0x6330a553fc93768f612722bb8c2ec78ac90b3bbc
(9) 0x5aeda56215b167893e80b4fe645ba6d5bab767de

Private Keys:
(0) c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3
(1) ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f
(2) 0dbbe8e4ae425a6d2687f1a7e3ba17bc98c673636790f1b8ad91193c05875ef1
(3) c88b703fb08cbea894b6aeff5a544fb92e78a18e19814cd85da83b71f772aa6c
(4) 388c684f0ba1ef5017716adb5d21a053ea8e90277d0868337519f97bede61418
(5) 659cbb0e2411a44db63778987b1e22153c086a95eb6b18bdf89de078917abc63
(6) 82d052c865f5763aad42add438569276c00d3d88a2d062d36b2bae914d58b8c8
(7) aa3680d5d48a8283413f7a108367c7299ca73f553735860a87b08f39395618b7
(8) 0f62d96d6675f32685bbdb8ac13cda7c23436f63efbb9d07700d8669ff12b7c4
(9) 8d5366123cb560bb606379f90a0bfd4769eecc0557f1b362dcae9012b548b1e5

Mnemonic: candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

⚠ Important ⚠ : This mnemonic was created for you by Truffle. It is not secure.
Ensure you do not use it on production blockchains, or else you risk losing funds.

truffle(develop)> █
```

Figure 4: Command: truffle develop



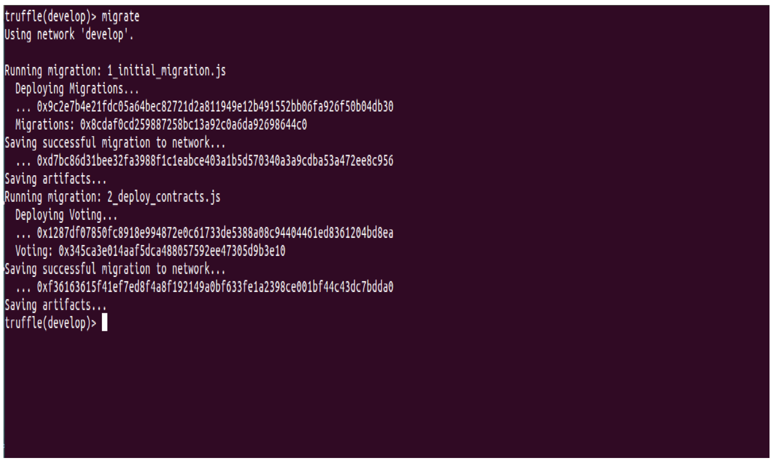
```
truffle(develop)> compile
Compiling ./contracts/Migrations.sol...
Compiling ./contracts/Voting.sol...

Compilation warnings encountered:

/home/grv/Desktop/eth-voting-dapp (copy)/contracts/Migrations.sol:11:3: Warning: Defining constructors as functions with the same name as the c
ontract is deprecated. Use "constructor(...) { ... }" instead.
  function Migrations() {
  ^ (Relevant source part starts here and spans across multiple lines).
,/home/grv/Desktop/eth-voting-dapp (copy)/contracts/Voting.sol:16:9: Warning: Defining constructors as functions with the same name as the cont
ract is deprecated. Use "constructor(...) { ... }" instead.
        function Voting()public {
        ^ (Relevant source part starts here and spans across multiple lines).
,/home/grv/Desktop/eth-voting-dapp (copy)/contracts/Voting.sol:65:13: Warning: Invoking events without "emit" prefix is deprecated.
            AddedCandidate(candidateID);
            ^--------------------------^
,/home/grv/Desktop/eth-voting-dapp (copy)/contracts/Migrations.sol:11:3: Warning: No visibility specified. Defaulting to "public".
  function Migrations() {
  ^ (Relevant source part starts here and spans across multiple lines).
,/home/grv/Desktop/eth-voting-dapp (copy)/contracts/Migrations.sol:15:3: Warning: No visibility specified. Defaulting to "public".
  function setCompleted(uint completed) restricted {
  ^ (Relevant source part starts here and spans across multiple lines).
,/home/grv/Desktop/eth-voting-dapp (copy)/contracts/Migrations.sol:19:3: Warning: No visibility specified. Defaulting to "public".
  function upgrade(address new_address) restricted {
  ^ (Relevant source part starts here and spans across multiple lines).

Writing artifacts to ./build/contracts

truffle(develop)> █
```

Figure 5: Command: compile

4.2.3 Compilation

After completing the previous step, we enter the truffle console. In this step, we compile the smart contracts and create the connected JSON files. Any errors or warnings during compilation are also shown in this stage. The command used, as shown in Figure 5, is:

```
compile
```



```
truffle(develop)> migrate
Using network 'develop'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0x9c2e7b4e21fdc05a64bec82721d2a811949e12b491552bb06fa926f50b04db30
  Migrations: 0x8cdaf0cd259887258bc13a92c0a6da92698644c0
Saving successful migration to network...
  ... 0xd7bc86d31bee32fa3988f1c1eabce403a1b5d570340a3a9cdba53a472ee8c956
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying Voting...
  ... 0x1287df07850fc8918e994872e0c61733de5388a08c94404461ed8361204bd8ea
  Voting: 0x345ca3e014aaf5dca488057592ee47305d9b3e10
Saving successful migration to network...
  ... 0xf36163615f41ef7ed8f4a8f192149a0bf633fe1a2398ce001bf44c43dc7bdda0
Saving artifacts...
truffle(develop)> 
```

Figure 6: Command: migrate

4.2.4 Migration

In this step, some pre-installed scripts are deployed which are required for deploying the smart contracts written. The command used, as shown in Figure 6, is:

```
migrate
```

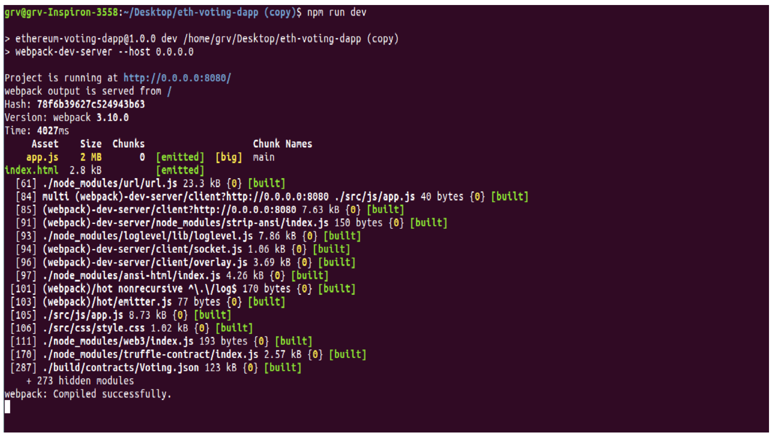
4.2.5 Deploy Web Server

Finally, in the last step, the web server is deployed. This launches the application, which can be accessed by any device in the same network. The command used, as shown in Figure 7, is:

```
npm run dev
```

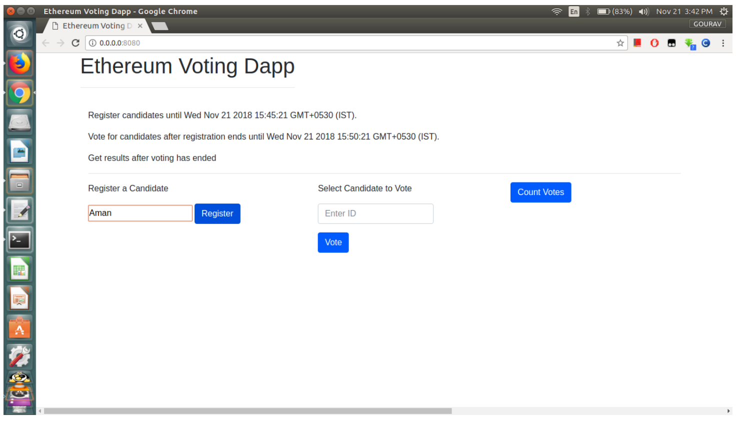
4.3 Result

After completing all the required steps, the e-voting application is ready to use. The following basic functionalities are provided by the application:



```
grv@grv-Inspiron-3558:~/Desktop/eth-voting-dapp (copy)$ npm run dev

> ethereum-voting-dapp@1.0.0 dev /home/grv/Desktop/eth-voting-dapp (copy)
> webpack-dev-server --host 0.0.0.0

Project is running at http://0.0.0.0:8080/
webpack output is served from /
Hash: 78f6b39627c524943b63
Version: webpack 3.10.0
Time: 4027ms
     Asset    Size  Chunks                    Chunk Names
    app.js    2 MB       0  [emitted]  [big]  main
index.html  2.8 kB          [emitted]
  [61] ./node_modules/url/url.js 23.3 kB {0} [built]
  [84] multi (webpack)-dev-server/client?http://0.0.0.0:8080 ./src/js/app.js 40 bytes {0} [built]
  [85] (webpack)-dev-server/client?http://0.0.0.0:8080 7.63 kB {0} [built]
  [91] (webpack)-dev-server/node_modules/strip-ansi/index.js 150 bytes {0} [built]
  [93] ./node_modules/loglevel/lib/loglevel.js 7.86 kB {0} [built]
  [94] (webpack)-dev-server/client/socket.js 1.06 kB {0} [built]
  [96] (webpack)-dev-server/client/overlay.js 3.69 kB {0} [built]
  [97] ./node_modules/ansi-html/index.js 4.26 kB {0} [built]
 [101] (webpack)/hot nonrecursive ^\.\/log$ 170 bytes {0} [built]
 [103] (webpack)/hot/emitter.js 77 bytes {0} [built]
 [105] ./src/js/app.js 8.73 kB {0} [built]
 [106] ./src/css/style.css 1.02 kB {0} [built]
 [111] ./node_modules/web3/index.js 193 bytes {0} [built]
 [170] ./node_modules/truffle-contract/index.js 2.57 kB {0} [built]
 [287] ./build/contracts/Voting.json 123 kB {0} [built]
    + 273 hidden modules
webpack: Compiled successfully.
```

Figure 7: Deploy Web Server, command: npm run dev



Ethereum Voting Dapp - Google Chrome

0.0.0.0:8080

Ethereum Voting Dapp

Register candidates until Wed Nov 21 2018 15:45:21 GMT+0530 (IST).

Vote for candidates after registration ends until Wed Nov 21 2018 15:50:21 GMT+0530 (IST).

Get results after voting has ended

Register a Candidate: Aman [Register]

Select Candidate to Vote: Enter ID [Vote]

[Count Votes]

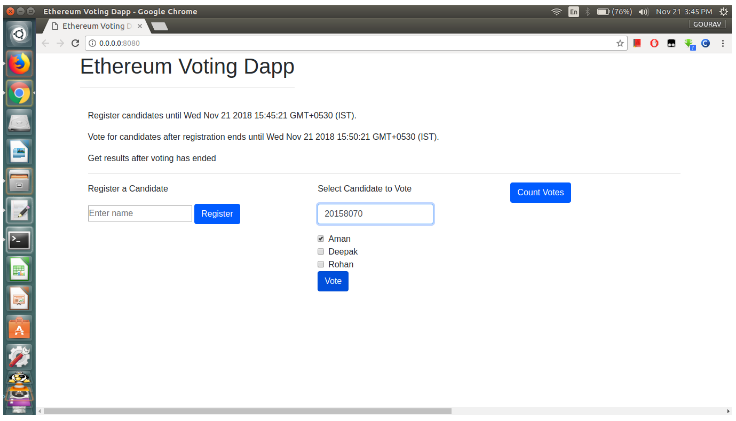
Figure 8: Register

4.3.1 Registration

Any person, who wants to participate as a candidate in the elections, can do so simply by submitting their name.

4.3.2 Voting

Voters can cast their vote and choose a candidate of their choice by selecting their name, and using their registration numbers. Currently, the voters are not verified.



Ethereum Voting Dapp

Register candidates until Wed Nov 21 2018 15:45:21 GMT+0530 (IST).

Vote for candidates after registration ends until Wed Nov 21 2018 15:50:21 GMT+0530 (IST).

Get results after voting has ended

Register a Candidate

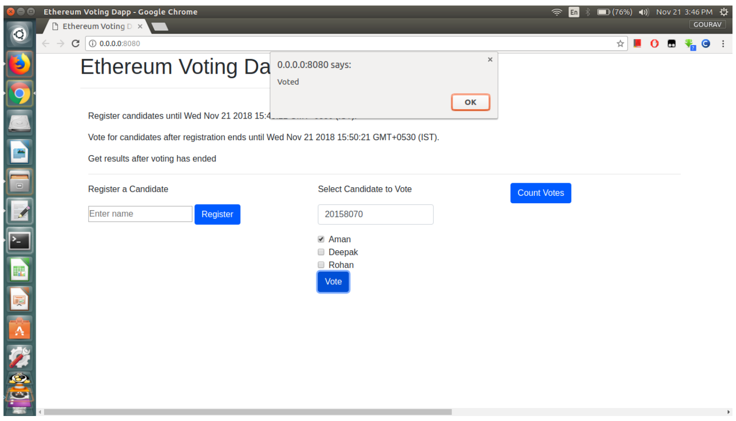
Enter name

Select Candidate to Vote

20158070

- [x] Aman
- [ ] Deepak
- [ ] Rohan

Figure 9: Vote



Ethereum Voting Da

0.0.0.0:8080 says:

Voted

Register candidates until Wed Nov 21 2018 15:4

Vote for candidates after registration ends until Wed Nov 21 2018 15:50:21 GMT+0530 (IST).

Get results after voting has ended

Register a Candidate

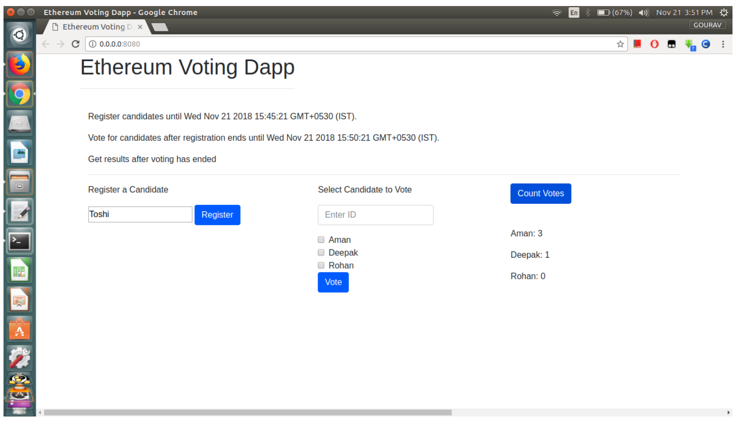
Enter name

Select Candidate to Vote

20158070

- [x] Aman
- [ ] Deepak
- [ ] Rohan

Figure 10: Confirmation of Vote


Ethereum Voting Dapp

Register candidates until Wed Nov 21 2018 15:45:21 GMT+0530 (IST).

Vote for candidates after registration ends until Wed Nov 21 2018 15:50:21 GMT+0530 (IST).

Get results after voting has ended

Register a Candidate

Toshi

Select Candidate to Vote

Aman: 3

Deepak: 1

Rohan: 0

Figure 11: Result Declaration

4.3.3 Result Declaration

After the elections have ended, any user can see the results easily.

4.4 Features Included

The following features are included in the application

1. Multiple People With Same Name

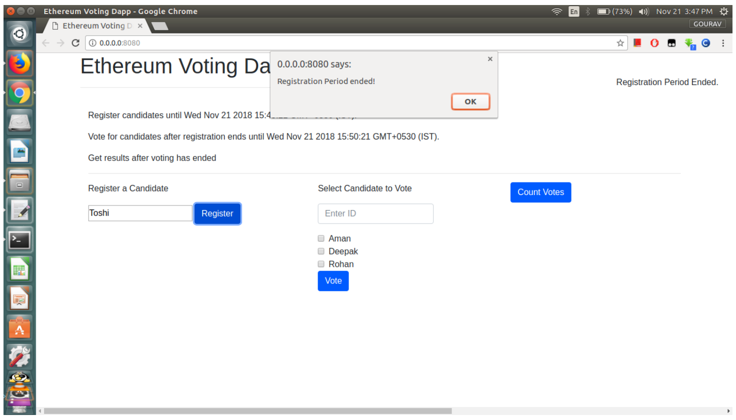
   It is possible for multiple people to have the same name. In cases such as these, the option of parties can be added so that candidates can be differentiated.

2. Association With Parties

   As hinted in the previous point, candidates can be associated with parties, if any. In the current version, the option of parties is available in the back end but is not shown in the front end since the project was made keeping in mind MNNIT's environment.

3. No Multi-Voting

   A single person cannot vote for multiple people. Although a quite basic requirement, this option is very useful for emphasizing that all voters are equal.



Ethereum Voting Da

0.0.0.0:8080 says:

Registration Period ended!

OK

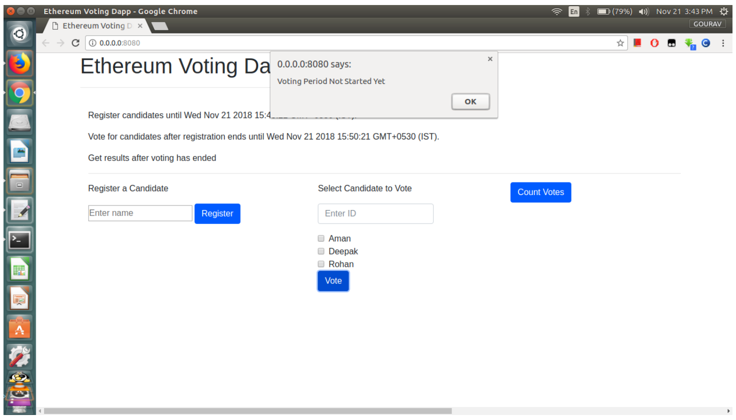
Registration Period Ended.

Register candidates until Wed Nov 21 2018 15:4

Vote for candidates after registration ends until Wed Nov 21 2018 15:50:21 GMT+0530 (IST).

Get results after voting has ended

Figure 12: End of Registration Period



Ethereum Voting Da

0.0.0.0:8080 says:

Voting Period Not Started Yet

OK

Register candidates until Wed Nov 21 2018 15:4

Vote for candidates after registration ends until Wed Nov 21 2018 15:50:21 GMT+0530 (IST).

Get results after voting has ended

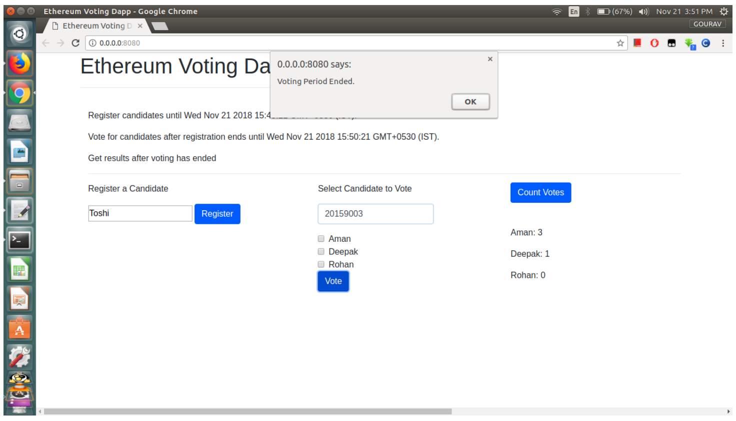
Figure 13: Voting Period Not Started

4. Vote Changing

   A person can change their casted vote, within specific time boundations. This option is included to account for votes cast by mistake.

5. Timings

   The option of timings is the highlight of the project. Just as in a real election, before voting starts, candidates are announced, which can't be done once voting starts, and results cannot be seen before voting has ended, and voting cannot be done before or after the voting process, this application also supports all these features.


Ethereum Voting Da

0.0.0.0:8080 says:

Voting Period Ended.

Register candidates until Wed Nov 21 2018 15:4

Vote for candidates after registration ends until Wed Nov 21 2018 15:50:21 GMT+0530 (IST).

Get results after voting has ended

Aman: 3

Deepak: 1

Rohan: 0

Figure 14: Voting Ended

4.5 Features Omitted

The following features have been excluded from the current version of the application to keep it simple, but can easily be added:

1. Voter registration and verification

   The feature of voter verification does not make any sense in a private blockchain. For verification, forgers are needed and private blockchains are used for testing purposes only.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Using blockchain and web architecture, here we propose an e-voting system that tries to overcome most of the shortcomings of the current ballot based voting system. It provides easy and efficient ways to vote and tally results. Although the system is cost efficient and fulfils the requirements suitably, there is always room for imporvements. Using this version of e-voting system, new and more efficient systems may be built, which can be a big leap towards truly fair elections.

5.2 Future Work

This application provides a base for future applications. There is a lot of potential in this application. The following improvements can be made in future versions of this application:

- Addition of more functionalities according to the need
- Cryptographic Security can be considered
- Private and Consortium blockchains can be used

Appendix A

Some Complex Proofs and Simple Results

A.1 Elliptic Curve Digital Signature Algorithm

ECDSA [34] is short for Elliptic Curve Digital Signature Algorithm. It's a process that uses an elliptic curve and a finite field to sign data in such a way that third parties can verify the authenticity of the signature while the signer retains the exclusive ability to create the signature. With bitcoin, the data that is signed is the transaction that transfers ownership.

ECDSA has separate procedures for signing and verification. Each procedure is an algorithm composed of a few arithmetic operations. The signing algorithm makes use of the private key, and the verification process makes use of the public key.

A.1.1 Elliptic Curves

An elliptic curve is represented algebraically as an equation of the form: $y^2 = x^3 + ax + b$

For a = 0 and b = 7 (the version used by bitcoin), it looks like Figure 15.

Elliptic curves have useful properties. For example, a non-vertical line intersecting two non-tangent points on the curve will always intersect a third point on the curve. A further property is that a non-vertical line tangent to the curve at one

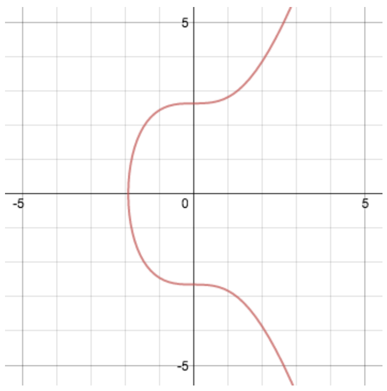


Figure 15: Elliptic Curve Example

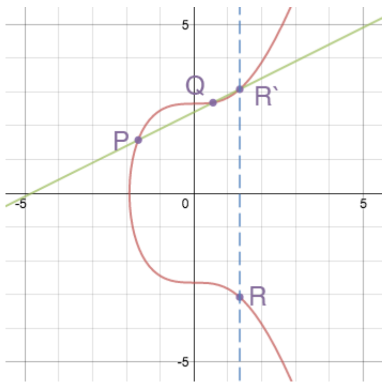


Figure 16: Point Addition

point will intersect precisely one other point on the curve.

We can use these properties to define two operations: point addition and point doubling.

Point addition, P + Q = R, is defined as the reflection through the x-axis of the third intersecting point R' on a line that includes P and Q. It's easiest to understand this using Figure 16.

Similarly, point doubling, P + P = R is defined by finding the line tangent to the point to be doubled, P, and taking reflection through the x-axis of the intersecting point R' on the curve to get R. See Figure 17 for an example.

Together, these two operations are used for scalar multiplication, R = a P, defined by adding the point P to itself a times. For example:

R = 7P

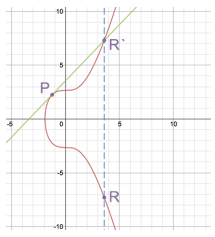


Figure 17: Point Doubling

R = P + (P + (P + (P + (P + (P + P)))))

The process of scalar multiplication is normally simplified by using a combination of point addition and point doubling operations. For example:

R = 7P

R = P + 6P

R = P + 2 (3P)

R = P + 2 (P + 2P)

Here, 7P has been broken down into two point doubling steps and two point addition steps.

A.1.2 Finite Fields

A finite field, in the context of ECDSA, can be thought of as a predefined range of positive numbers within which every calculation must fall. Any number outside this range wraps around so as to fall within the range.

The simplest way to think about this is calculating remainders, as represented by the modulus (mod) operator. For example, 9/7 gives 1 with a remainder of 2:

9 mod 7 = 2

Here our finite field is modulo 7, and all mod operations over this field yield a result falling within a range from 0 to 6.

A.1.3 Relation Between ECDSA and Finite Fields

ECDSA uses elliptic curves in the context of a finite field, which greatly changes their appearance but not their underlying equations or special properties. The same equation plotted above, in a finite field of modulo 67, looks like this:

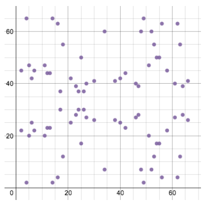


Figure 18: Elliptic Curve In Finite Field

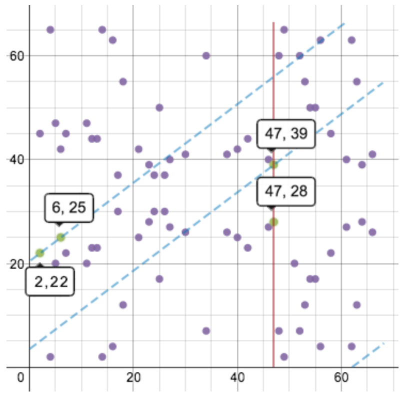


Figure 19: Point Addition In Finite Field

It's now a set of points, in which all the x and y values are integers between 0 and 66. Note that the curve still retains its horizontal symmetry.

Point addition and doubling are now slightly different visually. Lines drawn on this graph will wrap around the horizontal and vertical directions, just like in a game of Asteroids, maintaining the same slope. So adding points (2, 22) and (6, 25) looks like this:

The third intersecting point is (47, 39) and its reflection point is (47, 28).

A.1.4 ECDSA And Bitcoin

A protocol such as bitcoin selects a set of parameters for the elliptic curve and its finite field representation that is fixed for all users of the protocol. The parameters include the equation used, the prime modulo of the field, and a base point that falls on the curve. The order of the base point, which is not independently selected but is a function of the other parameters, can be thought of graphically as the number of times the point can be added to itself until its slope is infinite, or a vertical line.

The base point is selected such that the order is a large prime number.

Bitcoin uses very large numbers for its base point, prime modulo, and order. In fact, all practical applications of ECDSA use enormous values. The security of the algorithm relies on these values being large, and therefore resistance to brute force attack or reverse engineering.

In the case of bitcoin:

Elliptic curve equation:

$$y^2 = x^3 + 7$$

$$Primemodulo = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1 =$$

FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F

Base point = 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

Order = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141

Who chose these numbers, and why? A great deal of research, and a fair amount of intrigue, surrounds the selection of appropriate parameters. After all, a large, seemingly random number could hide a backdoor method of reconstructing the private key. In brief, this particular realization goes by the name of secp256k1 and is part of a family of elliptic curve solutions over finite fields proposed for use in cryptography.

A.1.5 Private Keys And Public Keys

With these formalities out of the way, we are now in a position to understand private and public keys and how they are related. In ECDSA, the private key is an unpredictably chosen number between 1 and the order. The public key is derived from the private key by scalar multiplication of the base point a number of times equal to the value of the private key. Expressed as an equation:

public key = private key * base point

This shows that the maximum possible number of private keys (and thus bitcoin addresses) is equal to the order.

In a continuous field we could plot the tangent line and pinpoint the public key on the graph, but there are some equations that accomplish the same thing in the context of finite fields. Point addition of p + q to find r is defined component-wise as follows:

$$c = \frac{(q_y - p_y)}{(q_x - p_x)}$$

$$r_x = c^2 p_x - q_x$$

$$r_y = c(p_x - r_x) - p_y$$

And point doubling of p to find r is as follows:

$$c = \frac{(3p_x^2 + a)}{2p_y}$$

$$r_x = c22p_x$$

$$r_y = c(p_x - r_x) - p_y$$

In practice, computation of the public key is broken down into a number of point doubling and point addition operations starting from the base point.

Let's run a back of the envelope example using small numbers, to get an intuition about how the keys are constructed and used in signing and verifying. The parameters we will use are:

Equation:

$$y^2 = x^3 + 7$$

(which is to say, a = 0 and b = 7)

Prime Modulo: 67

Base Point: (2, 22)

Order: 79

Private key: 2

First, let's find the public key. Since we have selected the simplest possible private key with value = 2, it will require only a single point doubling operation from the base point. The calculation looks like this:

$$c = \left(\frac{3 * 22 + 0}{2 * 22}\right) mod 67$$

$$c = \left(\frac{3 * 4}{44}\right) mod 67$$

$$c = \left(\frac{12}{44}\right) mod 67$$

Here we have to pause for a bit of sleight-of-hand: how do we perform division in the context of a finite field, where the result must always be an integer? We have to multiply by the inverse, which space does not permit us to define here. In the case at hand, we simply state that:

$$44^{-1} = 32$$

Moving right along:

$$c = 12 * 32 mod 67$$

$$c = 384 mod 67$$

$$c = 49$$

$$r_x = (49^2 - 2 * 2) mod 67$$

$$r_x = (24014) mod 67$$

$$r_x = 2397 mod 67$$

$$r_x = 52$$

$$r_y = (49 * (252) 22) mod 67$$

$$r_y = (49 * (-50) 22) mod 67$$

$$r_y = (-245022) mod 67$$

$$r_y = (-2472) mod 67$$

$$r_y = 7$$

Our public key thus corresponds to the point (52, 7). All that work for a private key of 2!

This operation going from private to public key is computationally easy in comparison to trying to work backwards to deduce the private key from the public key, which while theoretically possible is computationally infeasible due to the large parameters used in actual elliptic cryptography.

Therefore, going from the private key to the public key is by design a one-way trip.

As with the private key, the public key is normally represented by a hexadecimal string. But wait, how do we get from a point on a plane, described by two numbers, to a single number? In an uncompressed public key the two 256-bit numbers representing the x and y coordinates are just stuck together in one long string. We can also take advantage of the symmetry of the elliptic curve to produce a compressed public key, by keeping just the x value and noting which half of the curve the point is on. From this partial information we can recover both coordinates.

A.1.6 Signing Data With Private Key

Now that we have a private and public key pair, let's sign some data!

The data can be of any length. The usual first step is to hash the data to generate a number containing the same number of bits (256) as the order of the curve. Here, for the sake of simplicity, we'll skip the hashing step and just sign the raw data z. We'll call G the base point, n the order, and d the private key. The recipe for signing is as follows:

1. Choose some integer k between 1 and n - 1.
2. Calculate the point (x, y) = k * G, using scalar multiplication.
3. Find r = x mod n. If r = 0, return to step 1.
4. Find s = (z + r * d) / k mod n. If s = 0, return to step 1.

5. The signature is the pair (r, s)

As a reminder, in step 4, if the numbers result in a fraction (which in real life they almost always will), the numerator should be multiplied by the inverse of the denominator. In step 1, it is important that k not be repeated in different signatures and that it not be guessable by a third party. That is, k should either be random or generated by deterministic means that are kept secret from third parties. Otherwise it would be possible to extract the private key from step 4, since s, z, r, k and n are all known.

Let's pick our data to be the number 17, and follow the recipe. Our variables:

z = 17 (data)

n = 79 (order)

G = (2, 22) (base point)

d = 2 (private key)

Pick a random number:

k = rand(1, n - 1)

k = rand(1, 79 - 1)

k = 3

Calculate the point. This is done in the same manner as determining the public key, but for brevity let's omit the arithmetic for point addition and point doubling.

(x, y) = 3G

(x, y) = G + 2G

(x, y) = (2, 22) + (52, 7)

(x, y) = (62, 63)

x = 62

y = 63

Find r:

r = x mod n

r = 62 mod 79

r = 62

Find s:

s = (z + r * d) / k mod n

s = (17 + 62 * 2) / 3 mod 79

s = (17 + 124) / 3 mod 79

s = 141 / 3 mod 79

s = 47 mod 79

s = 47

Note that above we were able to divide by 3 since the result was an integer. In real-life cases we would use the inverse of k (like before, we have hidden some gory details by computing it elsewhere):

s = (z + r * d) / k mod n

s = (17 + 62 * 2) / 3 mod 79

s = (17 + 124) / 3 mod 79

s = 141 / 3 mod 79

s = 141 * 3-1 mod 79

s = 141 * 53 mod 79

s = 7473 mod 79

s = 47

Our signature is the pair (r, s) = (62, 47). As with the private and public keys, this signature is normally represented by a hexadecimal string.

A.1.7 Verify Signature With Private Key

We now have some data and a signature for that data. A third party who has our public key can receive our data and signature, and verify that we are the senders.

With Q being the public key and the other variables defined as before, the steps for verifying a signature are as follows:

1. Verify that r and s are between 1 and n - 1.
2. Calculate w = s-1 mod n
3. Calculate u = z * w mod n
4. Calculate v = r * w mod n
5. Calculate the point (x, y) = uG + vQ

6. Verify that r = x mod n. The signature is invalid if it is not.

z = 17 (data)

(r, s) = (62, 47) (signature)

n = 79 (order)

G = (2, 22) (base point)

Q = (52, 7) (public key)

Verify that r and s are between 1 and n - 1.

r: $1 \leq 62 < 79$

s: $1 \leq 47 < 79$

Calculate w:

w = s-1 mod n

w = 47-1 mod 79

w = 37

Calculate u:

u = zw mod n

u = 17 * 37 mod 79

u = 629 mod 79

u = 76

Calculate v:

v = rw mod n

v = 62 * 37 mod 79

v = 2294 mod 79

v = 3

Calculate the point (x, y):

(x, y) = uG + vQ

Let's break down the point doubling and addition in uG and vQ separately.

uG = 76G

uG = 2(38G)

uG = 2(2(19G))

uG = 2(2(G + 18G))

uG = 2(2(G + 2(9G)))

uG = 2(2(G + 2(G + 8G)))

uG = 2(2(G + 2(G + 2(4G))))

uG = 2(2(G + 2(G + 2(2(2G)))))

Notice that by using the grouping trick we reduce 75 successive addition operations to just six operations of point doubling and two operations of point addition.

Working our way from the inside out:

uG = 2(2(G + 2(G + 2(2(2(2, 22))))))

uG = 2(2(G + 2(G + 2(2(52, 7)))))

uG = 2(2(G + 2(G + 2(25, 17))))

uG = 2(2(G + 2((2, 22) + (21, 42))))

uG = 2(2(G + 2(13, 44)))

uG = 2(2((2, 22) + (66, 26)))

uG = 2(2(38, 26))

uG = 2(27, 40)

uG = (62, 4)

And now for vQ:

vQ = 3Q

vQ = Q + 2Q

vQ = Q + 2(52, 7)

vQ = (52, 7) + (25, 17)

vQ = (11, 20)

Putting them together:

(x, y) = uG + vQ

(x, y) = (62, 4) + (11, 20)

(x, y) = (62, 63)

Clearly last step is the bulk of the work. For the final step,

Verify that r = x mod n

r = x mod n

62 = 62 mod 79

62 = 62

Our signature is valid!

References

[1] Stuart Haber and W. Scott Strenatta's work on Blockchain, https://cryptonews.com/news/a-key-insight-for-the-blockchain-came-in-1991-1895.htm

[2] Improving the Efficiency and Reliability of Digital Time-Stamping, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.71.4891rep=rep1type=pdf

[3] Satoshi Nakamoto, https://en.wikipedia.org/wiki/Satoshi_Nakamoto

[4] Bitcoin Blockchain Size Statistics, https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/

[5] Uganda Election Controversy, https://www.theguardian.com/global-development/2016/feb/22/ugandan-elections-polls-fraud-yoweri-museveni

[6] Kenya's Elections Shamed, https://edition.cnn.com/2017/10/25/africa/kenya-election/index.html

[7] 70 Years since the rigged elections of 1946,http://www.rri.ro/en_gb/70_years_since_the_rigged_electio
2555994

[8] BlockChain, https://en.wikipedia.org/wiki/Blockchain

[9] Bitcoin, https://bitcoin.org/

[10] Byzatine Fault Tolerance, http://pmg.csail.mit.edu/papers/osdi99.pdf

[11] Deutsche Bank, https://deutschebank.co.in/

[12] US Federal Reserve, https://www.federalreserve.gov/

[13] Barclays Bank, https://www.barclays.in/

[14] Citigroup, https://www.citigroup.com/citi/

[15] NASDAQ, https://www.nasdaq.com/

[16] Cryptocurrencies and blockchain - European Parliament, http://www.europarl.europa.eu/cmsdata/150761/TAX3%20Study%20on%20cryptocurrencies%20a

[17] Homomorphic Encryption, https://en.wikipedia.org/wiki/Homomorphic_encryption

[18] An E-voting Protocol Based on Blockchain, https://eprint.iacr.org/2017/1043.pdf

[19] David L. Chaum, Untraceable electronic mail, https://dl.acm.org/citation.cfm?id=358563

[20] Electronic Voting Service Using Block-Chain, https://pdfs.semanticscholar.org/cec9/ac40361d84bd5352b75dc80524438f2153b3.pdf

[21] Smart Contracts, https://en.wikipedia.org/wiki/Smart_contract

[22] Ethereum Project, https://www.ethereum.org/

[23] IBM, https://www.ibm.com/

[24] Web3, https://web3js.readthedocs.io/en/1.0/

[25] BitTorrent, https://www.bittorrent.com/

[26] Popcorn Time, https://popcorntime.sh/

[27] Bitmessage Wiki, https://bitmessage.org/

[28] Tor, https://www.torproject.org/

[29] Facebook, https://www.facebook.com/

[30] Solidity, https://solidity.readthedocs.io/en/v0.4.25/

[31] Node.JS, https://nodejs.org/en/

[32] Truffle Suite, https://truffleframework.com/

[33] MetaMask, https://metamask.io/

[34] Elliptic Curve Digital Signature Algorithm, https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm