Decentralised Applications Using Ethereum Blockchain

R. Aroul Canessane,
School of Computing, Sathyabama Institute of Science and Technology,
Chennai, Tamil Nadu
aroulcanessane@gmail.com

N.Srinivasan,
School of Computing, Sathyabama Institute of Science and Technology
Chennai, Tamil Nadu

Abinash Beuria,
School of Computing, Sathyabama Institute of Science and Technology
Chennai, Tamil Nadu
abinashbeuria1@gmail.com

Ashwini Singh,
School of Computing, Sathyabama Institute of Science and Technology,
Chennai, Tamil Nadu
ashwinidotx@gmail.com

B. Muthu Kumar,
Department of Computer Science and Engineering Chennai Institute Of Technology Chennai, Tamil Nadu.
aroulcanessane@gmail.com

***Abstract* — The concept of blockchain without a doubt is a revolutionary concept. It is the underlying Technology behind bitcoin and many more cryptocurrencies. Although the people's focus being only at blockchain as cryptocurrencies in everyday services to do payments online without the interference of a third party will try to replace the current method of cash which is a really slow and ancient method. Blockchain is a zero trust network and this makes it a very powerful tool for various services provided that people are ready to believe and invest in it. In the Ethereum world, the blockchain runs on smart contracts which are self-executing applications that come at a cost of security. This zero-trust network is capable of replacing many of the debated process or activities in our day to day life.One of our biggest concerns is an E-voting system which must be secure. Blockchain being an immutable and append only ledger will not allow for any tampering while also being fully transparent. In this paper, we have implemented and tested a sample e-voting app running as a smart contract for ethereum network using E-Wallets. After an election is held, eventually, the ethereum blockchain will hold the records of ballots and voters thus giving us a clear and trusty network where mishandling is to a minimum..**

Keywords — Blockchain, Ethereum, Bitcoin, Smart Contracts, Solidity, e-Voting.

I. INTRODUCTION

Blockchain Technology is becoming more and more evident in our day to day lives. More and more companies have started using blockchain as an underlying network for their daily transactions. When blockchain became popular, it was only used for handling payments using the cryptocurrency Bitcoin, but over the years, various studies have been carried upon and have started to suggest that blockchain can be used in many more areas.

In the ever-growing world, where several transactions need the approval of a third party vendor or can only be done in their presence is becoming a slowing factor in these negotiations. In blockchain, there is no need for a central authority to approve of these transactions or execute the operations. Blockchain is a peer to peer based network which is run by all the nodes that are participating. Because of this, not only we have a zero trust network but also all the structural information is kept within the blockchain network. The devices connected to the network are called nodes and they keep a copy of the blockchain and all of the transaction history in the form of a distributed ledger. To make a transaction on the blockchain, a node must have some incentive for the other nodes to trust this node. This incentive can be in the form of the blockchain's money; Ether gas in the case of Ethereum[2] blockchain. The gas value is required to execute, publish or make transactions on the Ethereum blockchain. Ethereum blockchain consists of smart contracts which reside in the data logic layer of the blockchain. Nodes can read other smart contracts and can read and execute them. Each smart contract needs to have a maximum cost of execution which lets the fellow nodes the know how much gas is required to execute the particular smart contract so that it gets uploaded to the network and be available to every other node.

As mentioned, with the blockchains unique function[8] and a distributed ledger, tampering without setting off an alarm is really tricky. A fork in the blockchain is referred to a change that occurs in the blockchain which causes it to split into two paths. This change can be because of a change in the blockchain's protocol or a forced forgery. Although forks can be accidental or intentional, which can happen because of various reasons, all the nodes agree to a consensus and try to fix the problem in the chain, this is called a hard fork where the participating nodes in the blockchain come to a collective conclusion and agree on making only one of the split nodes as the correct one while the other is discarded and all of its transactions are lost.

This type of power offered by blockchain is very suitable in various important government level or high security projects such as e-Voting. E-Voting[3] has seen many improvements in the past few years but has never been perfected. With the help of blockchain, we can make an immutable ledger that hopefully resolves some of these allegations on fraudulent voting and vote counts. Many countries such as Brazil, India and U.S.A. are one of the active users of the E.V.M. The people of these nations still don't really trust the system and cases of false counting and hijacking of E.V.M. booths is a big concern. This is where blockchain comes into place to make sure that these cases are handled.

II. MOTIVATION

Our motivation here is to provide a platform for people where they are able to completely trust the system, carry out the transactions and not be worried about miscounting of their votes, their choice and their decision remains unaltered and thus cannot be manipulated in any way. The platform that we are going to provide can be accessed from any device[9] running the blockchain node such as computers, servers, embedded systems or even mobile devices in the near future.

In the current fast-paced world where time is of the essence, present voting systems require people to be present at a designated place on a given time thereby bringing people out of their comfort zone and the process of waiting to vote is much more time consuming than the actual time required to vote. There are many occurrence where people from the different parties are present directly at the voting sites to manipulate or threaten to buy out the votes, candidates with criminal connections have been able to push others out of the race[10]. Even though the process of voting is supposed to be secret and therefore it is carried out in booths but in rural areas, many times people follow the voters into the booths and manipulate them into voting for their party.

To overcome the mishappenings mentioned above, we are proposing a solution where the voter feels secure enough to vote whichever candidates he or she feels like without having anyone monitoring them which conveys a message of security and safety that they wont be affected because of their choice. Apart from this, the voters have a freedom of voting according to their comfort time and zone set by them without having the trouble to be in a rush. They can do this directly from their own devices and sense of security.

The biggest disadvantage of normal voting systems that we are overcoming is the problem of transparency. In a real case scenario, we get to know the outcome only after the vote count is finished therefore, people do not have any idea about the actual count. In our model, we can monitor the count in real time.

III. IMPLEMENTATION

For our implementation, we decided to use Ethereum Blockchain because it is one of the widely used and open source platform for developing and deploying decentralised applications. Ethereum provides a wide range of services and solutions with its readily available development tools and smart contracts. A smart contract is a self executing program that runs on the blockchain. In order for it to get compiled and deployed on the blockchain, each node on the network then executes the contract in exchange for some ether. The currency required to execute a contract is called gas amount and varies from contract to contract.

All the nodes on the Ethereum blockchain operate in real time. This ensures that each and every transaction that happen is verified by all the nodes or no nodes at all. If there is a discrepancy at a node in the network, all the surrounding nodes drop the contract and create a fork in the network. This creates a divergence in the blockchain which is discarded. Blockchain maintains a ledger that is completely online hence, it cannot be tampered with. If anyone were to change it, the hoax node would get rejected.

This helps our e-Voting idea tremendously as it allows us to not be worried about the fraud count or multiple votes and instead provide a hassle free voting experience. The structure that we will be using here is similar to an object oriented approach. For us to make it an easy process, we need to make sure of a few things. First, we need transparency. Thanks to blockchain, all the records on the network are available to everyone. Second, we need to eliminate the problem of fake votes. For this, we ensure that every vote is real and is done by a real person on the blockchain. Also, a person should be able to vote only once. All of these issues can be resolved if we use blockchain in the backend as an underlying layer. Due to the immutable nature of the blockchain, if something is initialised, it can never be altered.

The language of choice will be solidity which is a programming language that runs directly on the blockchain. The smart contracts are written in solidity as well. Contracts are executed by all the nodes and the updated information is shared amongst other nodes after a regular interval. These contracts should be validated by at least 2 nodes to become activated. Even though Ethereum blockchain is free to use, it costs ether for nodes to execute a smart contract. This cost is referred to as gas. In Ethereum, gas varies based on the smart contract and its functions. Smart contracts also don't require proof of work as each node will perform transactions with the contract. Since Ethereum main network deals with real ethereum which is costly, we will be using our own ethereum test network called truffle network. Truffle is one of the widely used testing framework for ethereum which makes it easy to deploy and execute smart contracts on the ethereum blockchain.

It gives us an option to set up a private or a public blockchain network. Ganache provides a graphical environment for us to work with. It also gives us 10 accounts with 100 Ether each which makes it easy to deploy and perform transactions on the blockchain.

In our code, we have a contract Election which is similar to a class in C++ or Java. Election has all the components required in our smart contract. To define each candidate standing in the election, we are using a complex datatype struct which has the candidate's ID, his Name and his vote count. Because of the nature of solidity all these values are set to null or 0 be default.

Once we have declared the candidate, we need to map candidates to voters so that one voter can only vote for a single candidate. Then, we need to store all the voted accounts as well as number of candidates that are in the election. Finally, we need to check and give permission to vote for only those people who still have to vote. To accomplish this, we will use an event that will only allow a voter to vote for once.

```
// Store accounts that have voted
mapping(address => bool) public voters;
mapping(uint => Candidate) public candidates;

// Store Candidates Count
uint public candidatesCount;

// voted event
event votedEvent (
    uint indexed _candidateId
);
```

Fig 2: Code block to map voters, candidates and event.

Now that all the definitions are out of the way, we will be adding functions to out contract which will help the user to vote. But before we start to vote, we must add candidates. For this, we will be using a private function addCandidates() in our contract so that only we are able to add candidates. The addCandidate function will add a new candidate to the blockchain and will increment the no of candidates by one so it is easier for our api to list our all the candidates.

```
function addCandidate (string _name) private {
        candidatesCount ++;
        candidates[candidatesCount] = Candidate(
                        candidatesCount, _name, 0);
}
```

Fig 2: Code block for adding a candidate to the list.

We add a constructor method to our code to make it easier to call the addCandidate() method whenever it is executed by any of the nodes.

```
constructor () public {
        addCandidate("Ashwini Singh");
        addCandidate("Abinash Beuria");
}
```

Fig 3: Code block for defining a constructor

Lastly, we add the vote() function which will allow our voters to vote for their preferred candidate. To identify which candidate a voter is voting, we will read the candidate's unique ID which we declared in the beginning. When the user selects and votes for a candidate, he/she has used their voting privileges and cannot vote again. To make sure this happens, we check for it by using a special function for users address called msg.sender(_address). This function takes in the address of the person calling our function vote and then checks with our mapping to see if the user has already voted or not. If the user has already voted, the require() will make sure that the rest of the code is not executed at all and the user isn't allowed to vote twice.

This takes away a lot of power from the user which he/she might use to cast fake votes and increase a particular candidate's count. Before voting, the user can see the total number of votes each candidate has. After voting, the window disappears.

To carry out the voting, users need to pay a small gas amount. This gas payment can be done in multiple ways. One of the ways that we will be using here Metamask. It is an extension for Google Chrome or Mozilla Firefox. Metamask allows us to visit the distributed blockchain through our browser and run ethereum contracts without running the full ethereum node. This is perfect for our app here because we are only concerned with the voting app and not with the other heavy transactions on the blockchain. To vote using metamask, we must create an account and connect to the test network using our given address

.

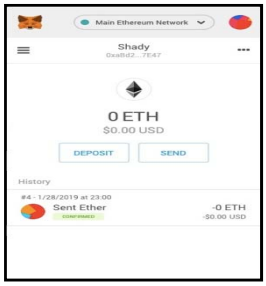
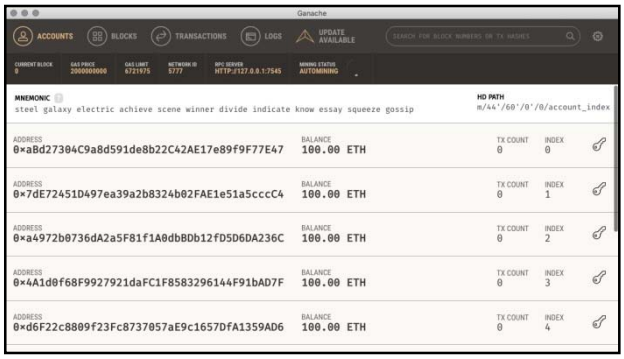
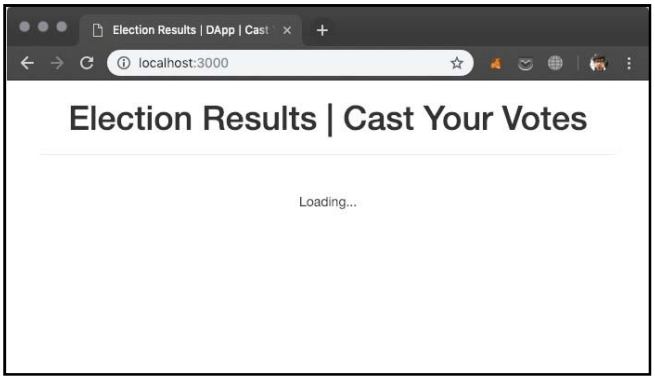


Fig 4: Metamask

The framework of our choice is Ganache which is





powered by truffle and has 10 accounts with 100ETH each to work with. To connect to this network, we start our Ganache then open the address provided in Ganache

Fig 5: Ganache

on our browser of choice. Once the page loads, we log in to our metamask account and connect to the test network we are running on.

Fig 6: Screenshot of our App without login.

In Fig 6, we see that our app is running but there is no option to vote. That's because the user hasn't logged in to their metamask account. This stops anonymous voting.

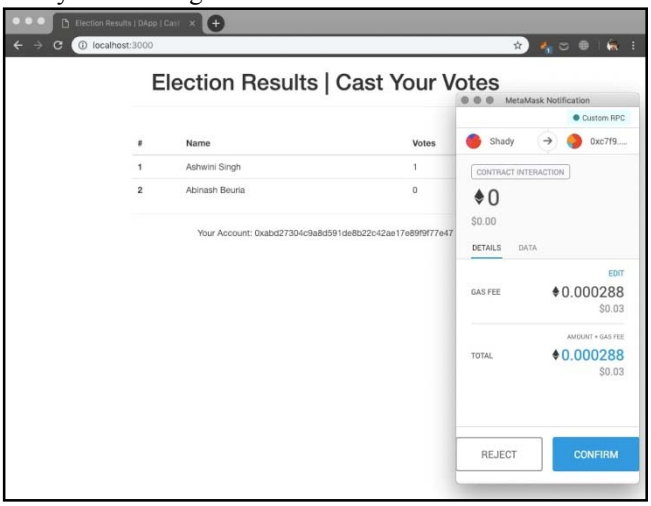


Fig 7: Logged in and about to vote using metamask.

Voting on the blockchain uses a fraction of our ether as gas amount. Once the gas is paid, the transaction occurs and all the other nodes get to know about this transaction as well so the voting occurs in real time and becomes difficult to tamper with..

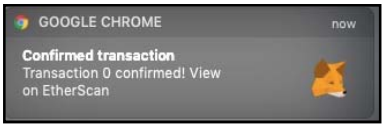


Fig 8: A successful transaction

Fig 9: Screenshot of the deployment process.

Since a user can vote only once, after voting, he/she doesn't get the interface to vote again but instead they get to see the results of the election. To make our app run, we migrate our project on the ethereum blockchain using truffle. After switching on Ganache, we have to reset the migrations that we deployed earlier. Then we run the entire process using the following code *npm run dev*. Once our app is running, we can see all the transactions happening on the blockchain at run time.

IV. CONCLUSION

Our main goal of decentralising the transactions happening with the voting system has been accomplished with the application created and we have been successful in securing the privacy as well as keeping it transparent to safeguard the decision of the users. We were also able to remove the disadvantages of time and location constraints by now allowing the users to vote from their own blockchain nodes at their own convenience.

The entire counting process of the votes is open to all to be monitored thus, reducing any chances of manipulations of the votes and the results are seen in real time.

ACKNOWLEDGEMENT

This paper wouldn't have been done without the support and availability of Ethereum as an open source blockchain which was made available by Vitalik Buterin.

REFERENCES

[1] S. Nakamoto, "Bitcoin: a peer to peer electronic cash system" : https://bitcoin.org
[2] Ethereum project: https://ethereum.org
[3] E-Voting, GitHub : https://github.com/topics/e-voting

[4] Truffle : https://truffleframework.com
[5] Ganache: https://truffleframework.com/ganache
[6] Shitang Yu,Kun Lu,Zhou Shao,Yingcheng Gou,Bo Zhang(2018)," A High Performance Blockchain Platform for Intelligent Devi" IEEE International Conference on Hot Information-Centric Networking.
[7] John Domingue ; Michelle Bachler ; Kevin Quick," Smart Blockchain Badges for Data Science Education"2018 IEEE Frontiers in Education Conference (FIE)
[8] Nir Kshetri ; Jeffrey Voas(2018)"Blockchain in Developing Countries" IT Professional (Volume: 20 , Issue: 2 , Mar./Apr. 2018)
[9] Nir Kshetri ; Jeffrey Voas-2018 "Blockchain-Enabled E-Voting", IEEE Software (Volume: 35 , Issue: 4 , July/August 2018)
[10] Nir Kshetri ; Jeffrey Voas-2018 "Blockchain-Enabled E-Voting", IEEE Software (Volume: 35 , Issue: 4 , July/August 2018)