CS 550 - Distributed Systems
Fall 2021
Group Project

By: Cem Taha Aker, Deniz Doğru, Anıl Elakaş

Title: An Online Voting System implementation with a Blockchain

Abstract

In today's internet landscape blockchain technology is increasing its share of distributed systems. This transformation is called Web 3**[10]**. Centralized networks are giving way to decentralized networks including the physical infrastructure of the internet. Other than some established blockchains which most people are aware of, there is a whole emerging ecosystem of interoperable blockchain networks. Our work represents an introduction to such systems by constructing a simple voting system using a blockchain to communicate and store transactions in a secure manner. We demonstrate this system using the Python programming language.

Contents

1. Introduction

1.1 System Independent Online/Offline Blockchains

A blockchain is a database which uses data structures called blocks to record transactions. The blocks contain hashes of the transactions that can be checked by participants in the network which also hold a copy of the blockchain. When a new transaction occurs, the participants check the transaction and the previous block hash before adding a new block. Networks differ from each other depending on the solution or algorithm they use in order to add a new block. This process can have different names depending on the network. Online refers to a continuous connection between participants.

Participants or nodes communicate changes in the blockchain with messages or file transfers. This communication method is also seen in a client-server or peer-to-peer network(p2p).

A distributed client-server or p2p model**[11]** for networking can be transformed into a blockchain by moving the consensus mechanism from the server to the clients thereby creating a decentralized network where each client can act as a server or in the case of voting systems such as Paxos and Raft**[9]**, a proposer.

In public blockchains anyone can review or validate transactions.

Section 1 of the project discusses blockchains. Section 2 gives details on background for centralized and decentralized systems. Section 3 explains the system architecture. Section 4 explains the system and test environment. Section 5 discusses the results.

1.2 Blockchain Resilience

Blockchains store a transaction history which is difficult to alter. Resilience is provided by reducing single point of failures and working similar to a p2p network.

1.2.1 Proof of Work Blockchains

Proof of Work(PoW) is a consensus mechanism in a blockchain to verify the integrity of each block by an individual node by protecting the overall architecture from malicious attacks and corrupted nodes. Although, in a private blockchain like our project, we assume that each individual node is honest (byzantine fault-tolerant) thus, what would the PoW serve? However, if the network expanded to include outside nodes to create a public blockchain, a PoW mechanism would have to be implemented so that trust is removed at the cost of additional computing resources to implement this mechanism.

1.2.2 Online Ballot Marking Problems

Online voting using blockchain architecture solves some of the problems created by a regular server-client connection, where the establishment is composed of HTML, javascript and JSON-based configuration files. In the chainless voting system, **[1]** online ballot marking works similar to our project. Such that a user registers to the system, enters credentials to log in and then proceeds to enter his/her vote to be saved on the sockets of the network. However in this design, the whole of the service architecture can be compromised through inside and outside attackers leading to biased election results. We aim to solve these security and scalability issues by implementing a blockchain server.

1.3 Solution with Consensus

Voting systems make use of the consensus mechanism for blockchain networks. This is crucial for the implementation of new code on the existing network. It is also necessary for a governance mechanism **[8]** for the nodes that take part in the network. The blockchain consensus protocols are the technology behind the security and performance of a valid blockchain which solves many of the problematic aspects of a classic client/server architecture.

There are five core components of a blockchain consensus protocol which facilitate discussion of the fault tolerance processes. Fault tolerance is the property that enables a system to continue operations in the events of failure of one or more faulty parts within the system's components. These components are namely, block proposal, block validation, information propagation, block finalization and incentive mechanism.

The goal of the blockchain consensus protocol is to ensure that all participating nodes, voters in our case, agree on a common network transaction history. **[8]** The consensus is represented in four sub titles of the consensus protocols which are termination, agreement, validity and integrity. In the termination step, a new user vote is either discarded or accepted to the chain. The agreement step requires serialization of blocks and transitions. The second block must carry the contents of the previous hash and the block after that must carry the contents of the previous block so on until all nodes agree to the sequence number. The validity requirement is similar to the termination counterpart as they represent the running voting system. Integrity requirement of the system fulfills the promise that all accepted transactions between honest nodes are consistent and all the accepted blocks are correctly generated and hash-chained in chronological order.

2. Background

2.1 Centralized Client/Server Systems

The client/server model has two components, a server and a client. These components interact through a network connection with a protocol**[11]**. They communicate with request and response messages. The internet as we know it is currently managed by such systems e.g. domain name system(DNS). Servers provide name resolution to addresses requested by clients. The servers process uniform resource locators(URL) to domain name by redirecting to top-level and second level domains.

2.2 Decentralized P2P Systems

An example for decentralized systems can be found in peer-to-peer networks or P2P. These relied on many client to client connections to find and deliver the intended request, such as a message, to a recipient. Skype was an early P2P communication protocol in the 2000's before its acquisition by Microsoft which then converted the network to a centralized system. File sharing has been a more resilient use case for P2P networks and these networks continue to operate.

2.3 Decentralized Blockchain Systems

Decentralization in the blockchain architecture refers to the transfer of control and decision-making from a centralized governance to a distributed network. Decentralized networks reduce the trust between peers. They alter the ability to put pressure or exert control in ways that could degrade the functionality of the network. Bitcoin is the first widely used successful blockchain and it is outlined in the paper by Satoshi Nakamoto. It is a p2p online payment system using a proof of work chain**[12].**

When building a technology solution such as online voting over networks, **[7]** typically three architectures are considered: centralized, distributed and decentralized. A blockchain technology makes use of decentralization which achieves management of access to resources in application to result in a greater service. Optimally, decentralized blockchain benefits from a trustless environment, improved data reliability by keeping record of each transaction hash in the ledger and resource distribution to reduce the likelihood of catastrophic failures such as votes being lost or voter information getting corrupted.

The block data structure alone does not provide security and reliability to the system but the decentralized nature of it does.

3. System Architecture

The system architecture of our project can be seen in figure 3.2.1. The application layer consists of main.py, gui.py and blockchain.py.

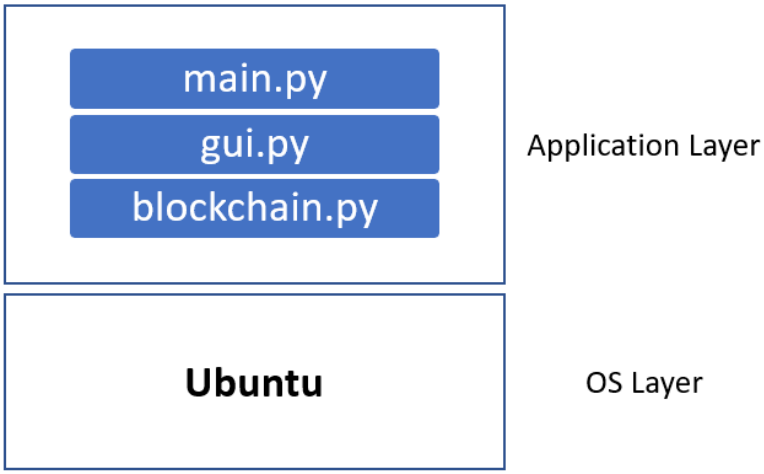

Figure 3.1 - System Architecture - OS and Application Layer

main.py - This file starts peer-to-peer connection when it is run from the terminal. The first peer acts like a server and the rest act like a client. However, if the server disconnects, one of the clients will act like a server. That means there is no centralized structure. After initiating the connection, this call the ***gui.py*** and GUI opens.

gui.py - This file opens the GUI on the screen. First, the home screen opens. There is a welcome message, Login and Exit buttons. When the Login button is clicked, the login screen opens. When the Exit button is clicked, GUI will be closed. In the login screen there are username textbox, password textbox, Login button and Close the Window button. When the user provides a username and password and if the values are valid, the Success Message and Vote button will be displayed. When the Vote button is clicked, the Vote Screen will be open. In the Vote Screen, users will see the pictures and Vote buttons for each of the candidates. When

one of the Vote buttons is clicked, *the addBlock* function from the ***blockchain.py*** will be called with “username,password,vote” data. After voting is done, the user can close the GUI.

blockchain.py - This file adds the voting data (“username,password,vote”) to the blockchain. If it is the first vote, it initiates the blockchain with the Geneysis block. The data is hashed and added to the blockchain. Each block contains a hash and the previous block’s hash. After user vote, Blockchain can be seen from blockchain.txt which is in the main directory.

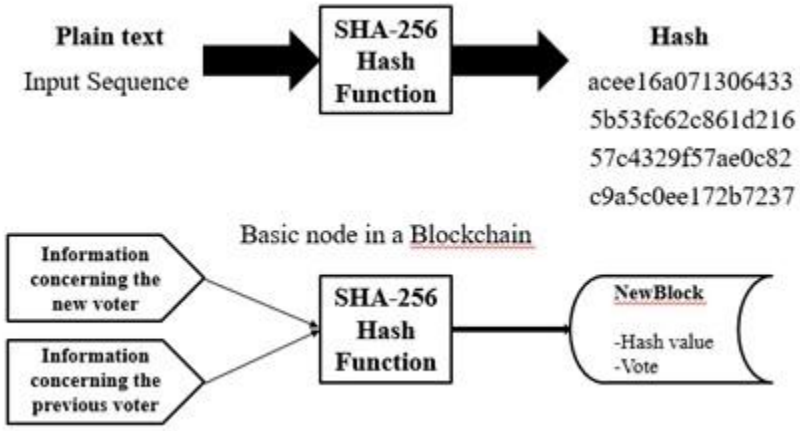

Figure 3.2 - SHA-256 Hash Algorithm

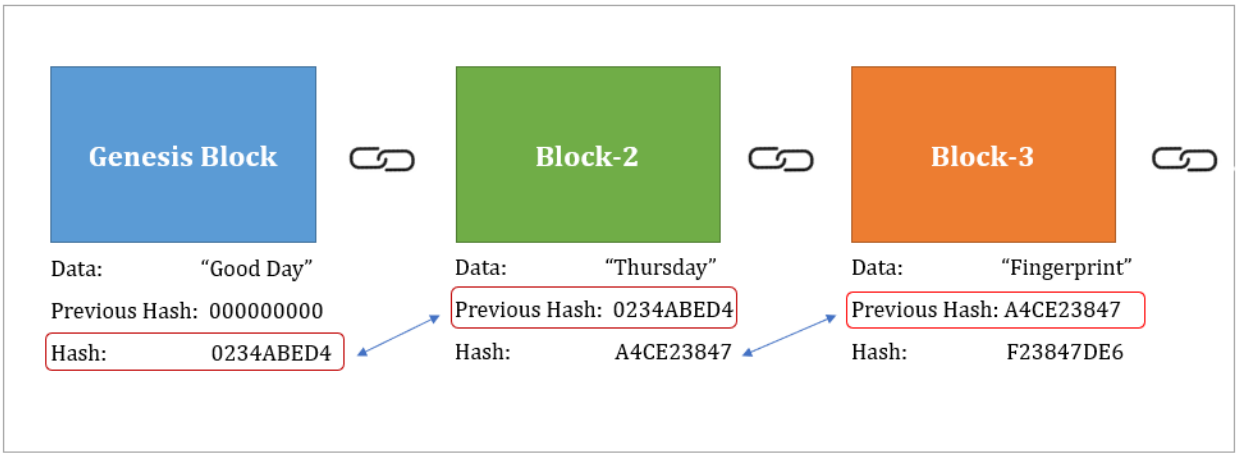

Figure 3.3 - Blocks in a blockchain linked cryptographically through hash

4. System and Test Environment Setup

4.1 System Setup

Virtual machine(VM) running Ubuntu 20.04 LTS Linux 5.8 Kernel on VMware Workstation Player 15.
The host machine OS is Windows 10 Home build 19041.867 and has Intel Core i7-7700K @4.20 GHz (4 cores, 8 logical processors) as the central processing unit. All operating systems are 64-bit.

Image showing system information for group member machine

```
anilelakas@LAPTOP-MTR34T9E:/mnt/c/Users/lenovo/Desktop/e-voting$ sudo lshw -short
[sudo] password for anilelakas:
H/W path  Device  Class      Description
=====================================
                  system     Computer
/0                bus        Motherboard
/0/0              memory     15GiB System memory
/0/1              processor  Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
/1        eth0    network    Ethernet interface
/2        eth1    network    Ethernet interface
/3        eth2    network    Ethernet interface
/4        wifi0   network    Ethernet interface
/5        wifi1   network    Ethernet interface
/6        wifi2   network    Ethernet interface
```

Python 3.x with python lib dependencies; socket for TCP connection,Tkinter for interface. To use tkinter, install it with the command “pip install tk”.
We used local-offline storage due to limited resources to record transactions, voter registration and results e.g. text file with .txt extension. To use an online system we would need a system to operate without downtime.
If a Linux operating system is not available, in a Windows machine Ubuntu terminal can be accessed with Windows Subsystem for Linux (WSL).

Tools used: Jupyter Notebook bundled with Anaconda 3, Python Shell, Google Collab

- Tkinter: Standard Python interface to the Tk GUI toolkit. Available on most Unix platforms as well as windows systems. Used in the implementation of the online voting interface.

- Sqlite3 : C-library providing a lightweight , self-contained ,file-based SQL database which comes bundled with Python. Does not need installation.

- Xming (required for Windows) : Xming is a display server for machines running a Windows operating system. Installation steps are available at README.txt provided in the original project folder.

4.2 Test Setup

Demonstration video provided in the original project folder. Launching Xming, running the main.py from the terminal, GUI, explanation about voters.txt, vote_history.txt, blockchain.txt and further explanation about the project components can be found in the demonstration video.

TESTING: The testing and the running of the program is explained in the steps below:

The user runs the main.py file in the selected python supported console/application. If the Xming application on windows is up and running, the server will successfully connect and the program graphical interface will launch.

Graphical Interface

- The first screen is named (Blockchain Voting), which is the toplevel screen of the program interface, it displays the label “Welcome to the Blockchain E-Voting Application”. Below there are two buttons which are login and exit.

- To login to the system, the user is given username-password combinations in the “voters.txt” file where they can add username-password combinations of their choice. This is the list of voters.

- Upon selection of login, another screen (Login) will appear on top. On the screen, there are two buttons available to login and close the window.
  If the user selects the button to close the window , the program will exit.
  If the user enters the correct username-password combinations and selects the button login, they will proceed to the next step.

- If the user enters combinations which are not available in the “voters.txt” file, then the program will display the message : “login not successful, you are not able to vote!”
- Else, the program will display the message: “Login successful. You can vote now!”

- After this step, the voting screen will be available with four candidates to choose from the screen. In this step, the user can either select the vote button assigned right in the middle of the candidate photos or they can choose to close the window which exits the application.

- When the user votes, another pop-up screen will appear with the message displaying: “Your vote is added to the blockchain successfully!”. Thus, the user vote is added to the blockchain and the voting process is complete.

- Lastly, If the same user tries to vote again, the server will display the error message : “You have already voted, you cannot vote again!”

5. Results

We obtained a blockchain containing votes for a single election. Every election is a different blockchain. We record the results for an election separately. When a new election starts, unless the previous blockchain is stored elsewhere, it is cleared and a new blockchain is initiated with a new genesis block.

Voters are determined before the election. Only these voters can vote in an election. This determines the amount of eligible voters i.e. max(voters). When a voter registers they are checked against the voting list to see if they are eligible to vote for any election. A voter can only vote once and if they try to vote again the vote is checked against the blockchain to see if it is a valid or duplicate vote.

An election can be completed before all eligible voters have voted or finished voting(before a vote is added as a block).

Although locally blocks seem unique to a client thus asynchronous, these blocks are added to the blockchain in the order of completed votes i.e. chronologically thereby synchronizing the transactions and creating a valid transaction history.

Similar to a simple p2p network running on a local host, clients are initiated by running separate command lines on a single machine. In the future we would like to add functionality for other machines to connect to the network.

6.Appendix

Figure 1 - Blockchain E-Voting Application- First screen after running main.py .

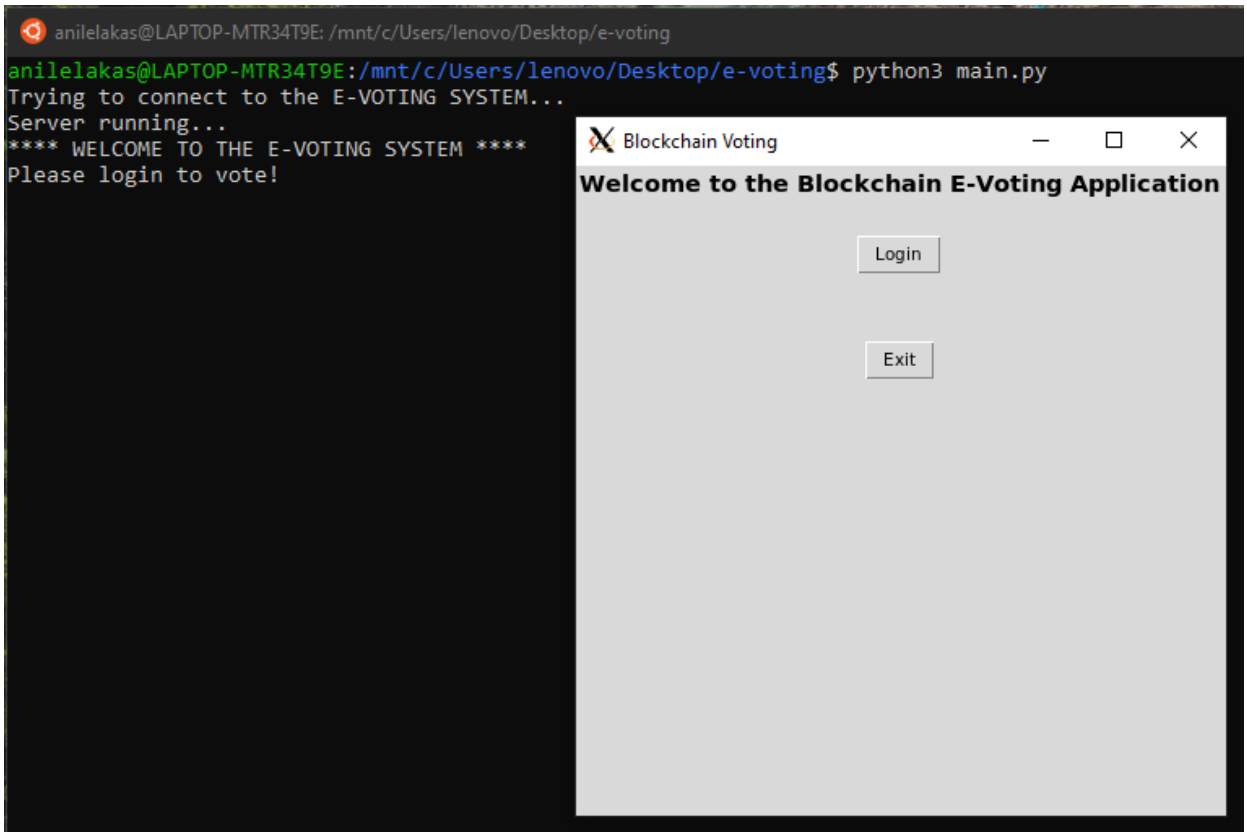

Figure 2 - Login Screen - Provide Username and Password.

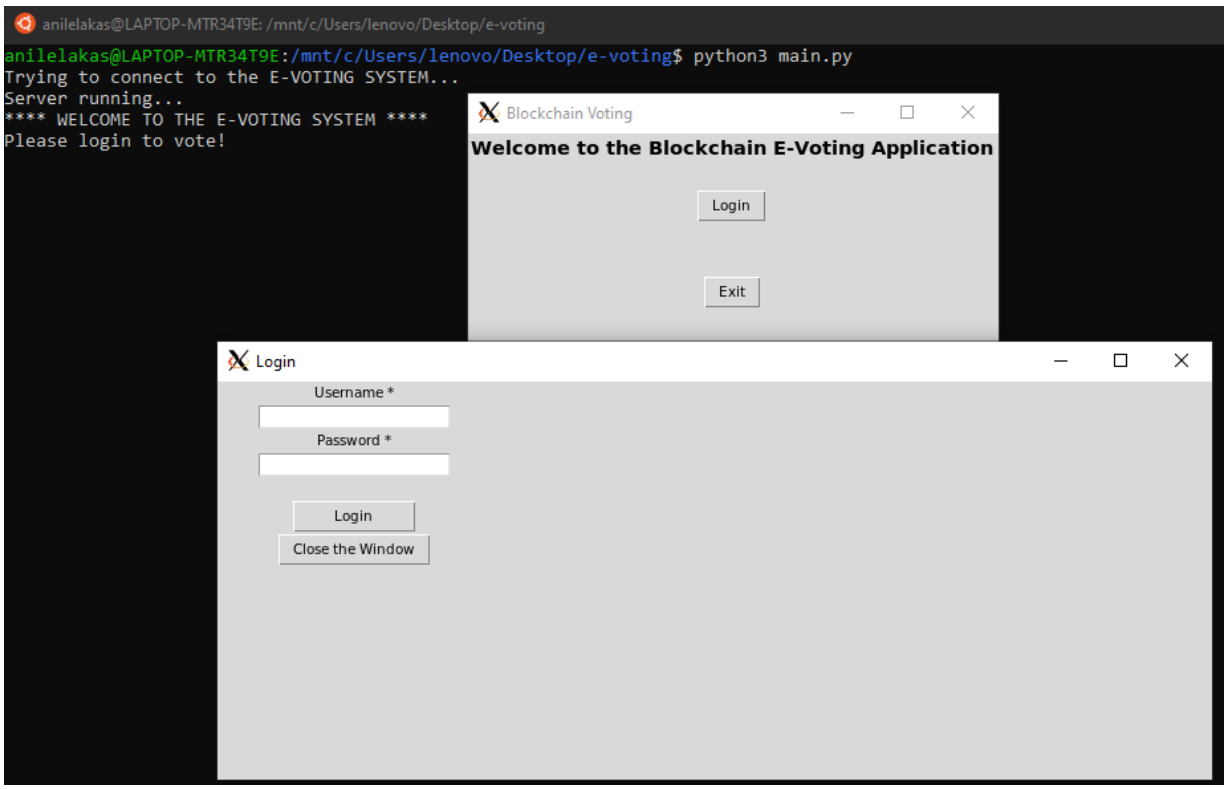

Figure 3 - Login Screen - Not Successful Login - Providing invalid username and password.

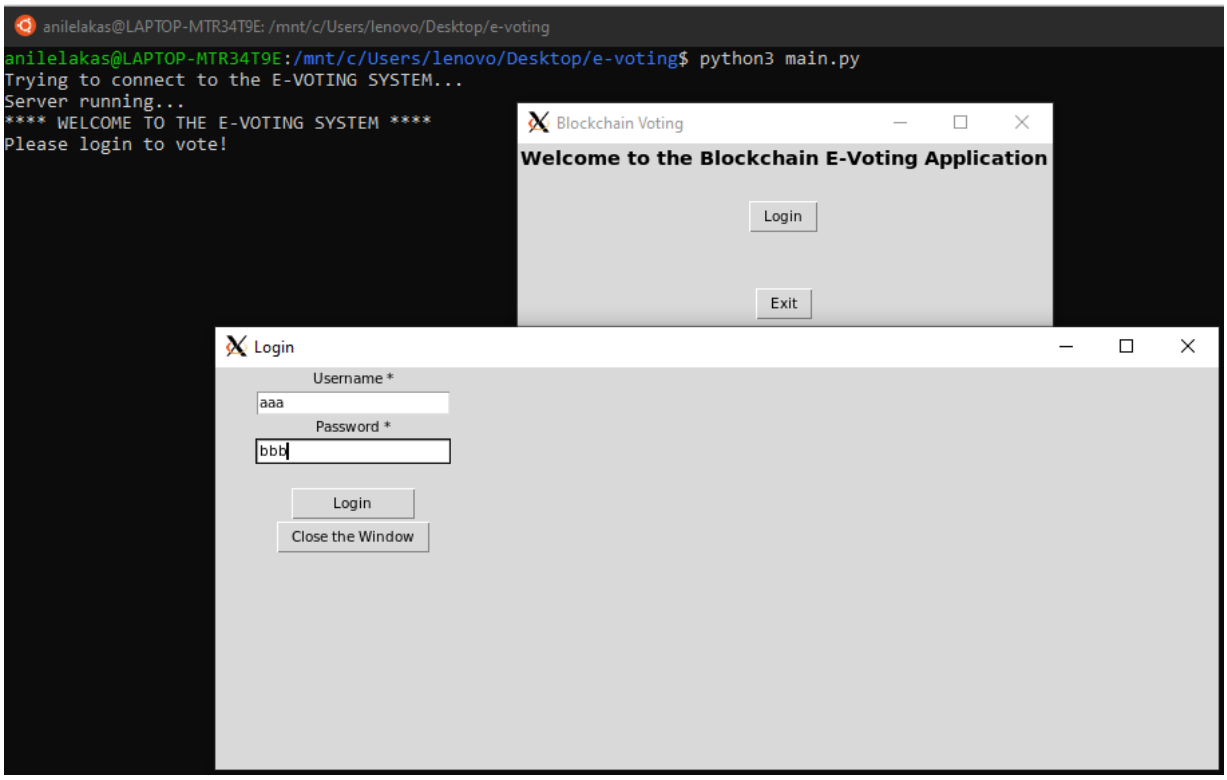

Figure 4 - Login Screen- Not Successful Login - Displaying Error Message.

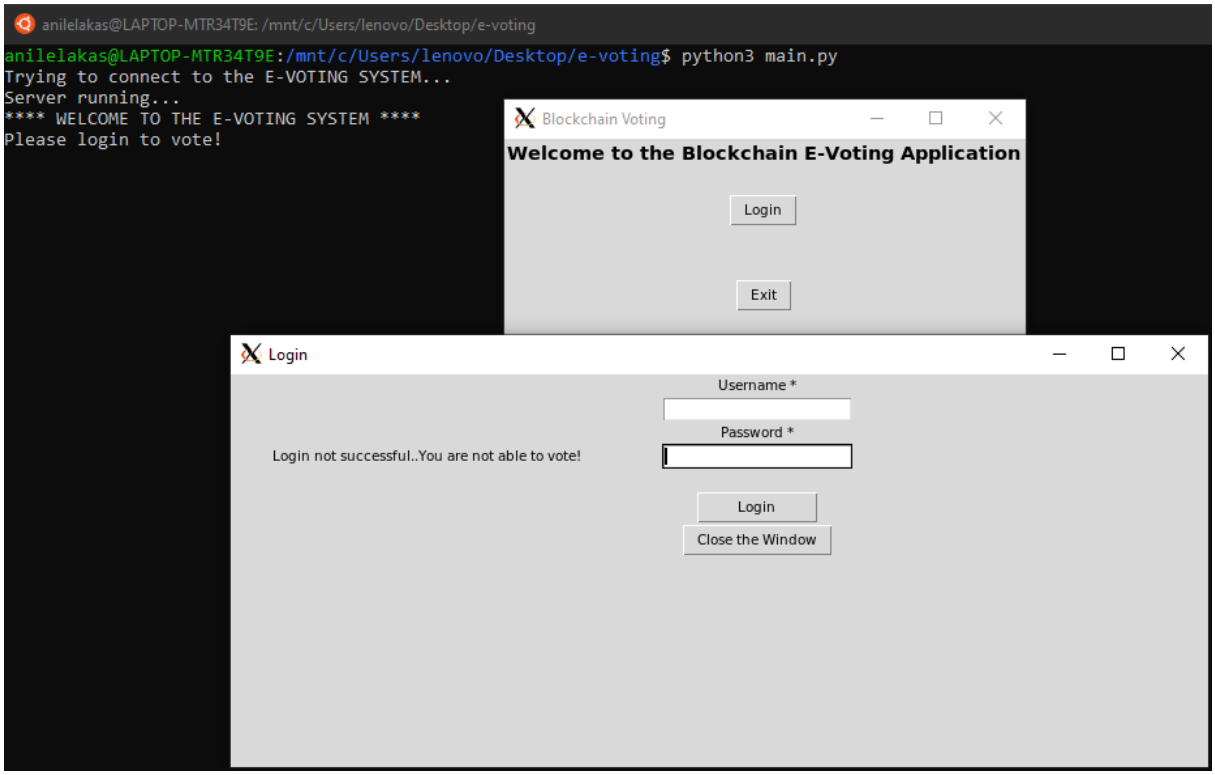

Figure 5 - Login Screen- Successful Login - Providing valid username and password information.

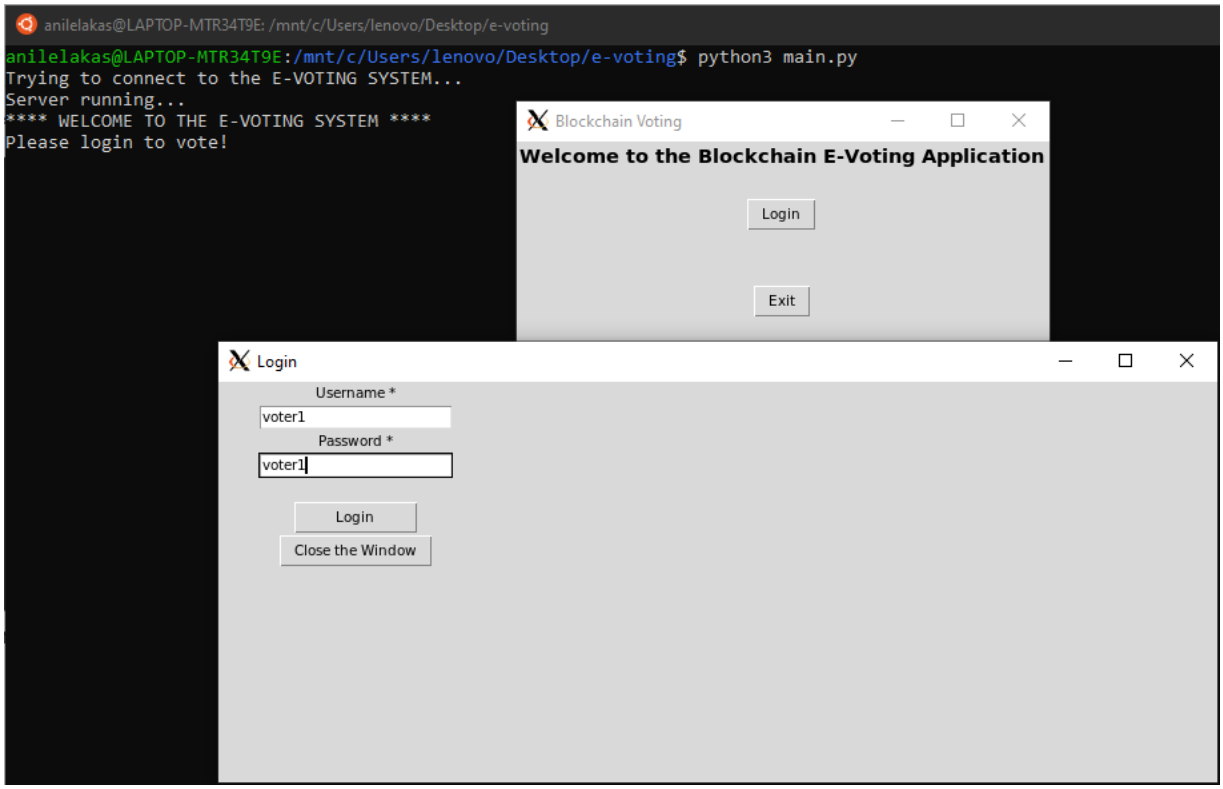

Figure 6 - Login Screen- Successful Login - Displaying Success Message.

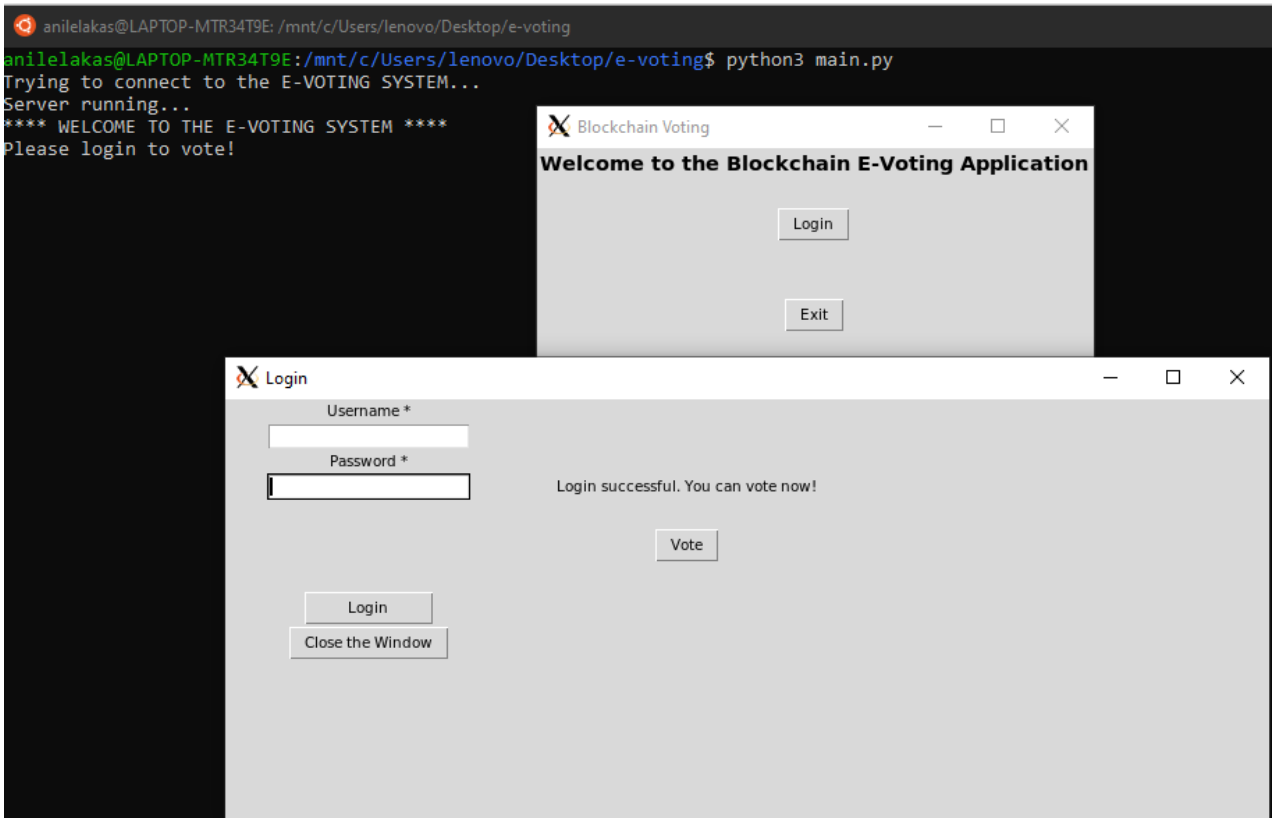

Figure 7 - Voting Screen - Vote for your candidates.

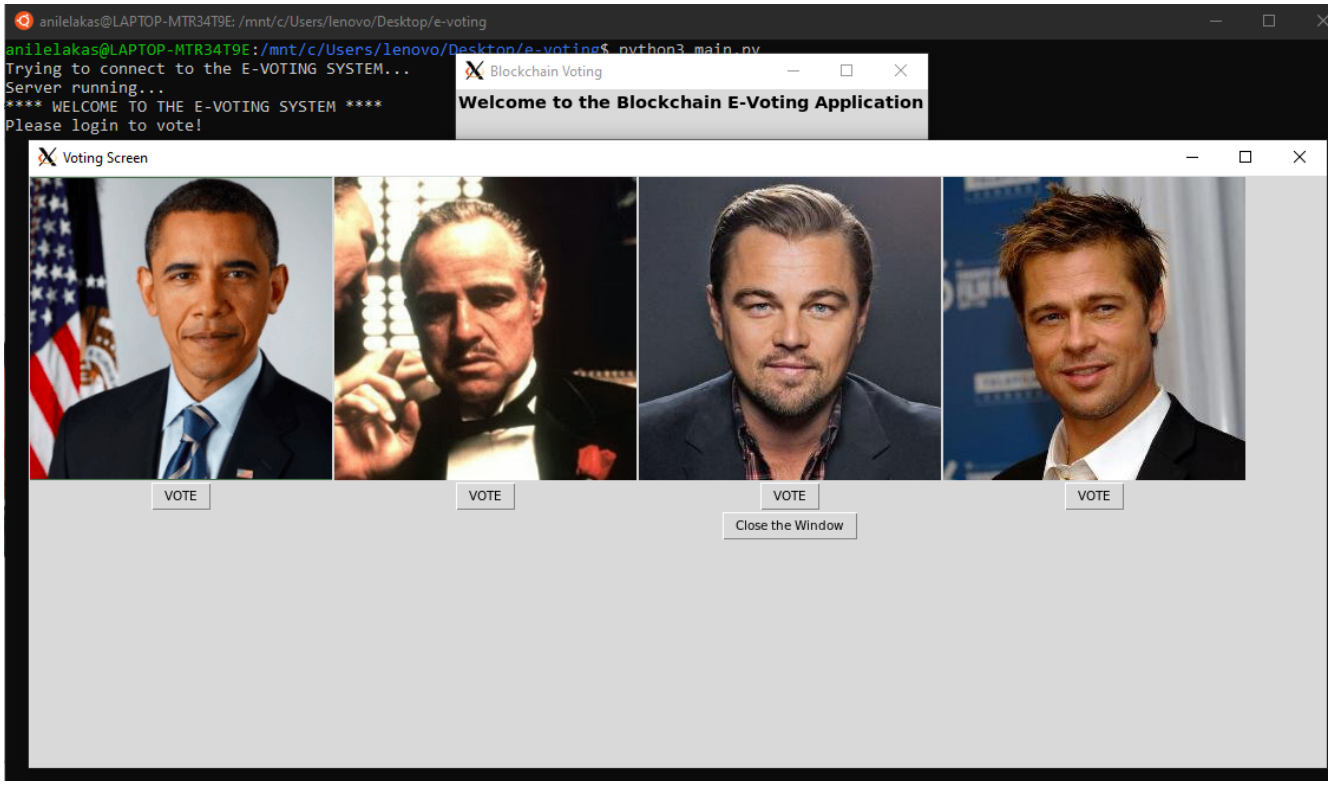

Figure 8 - Voting Screen - Successful Voting.

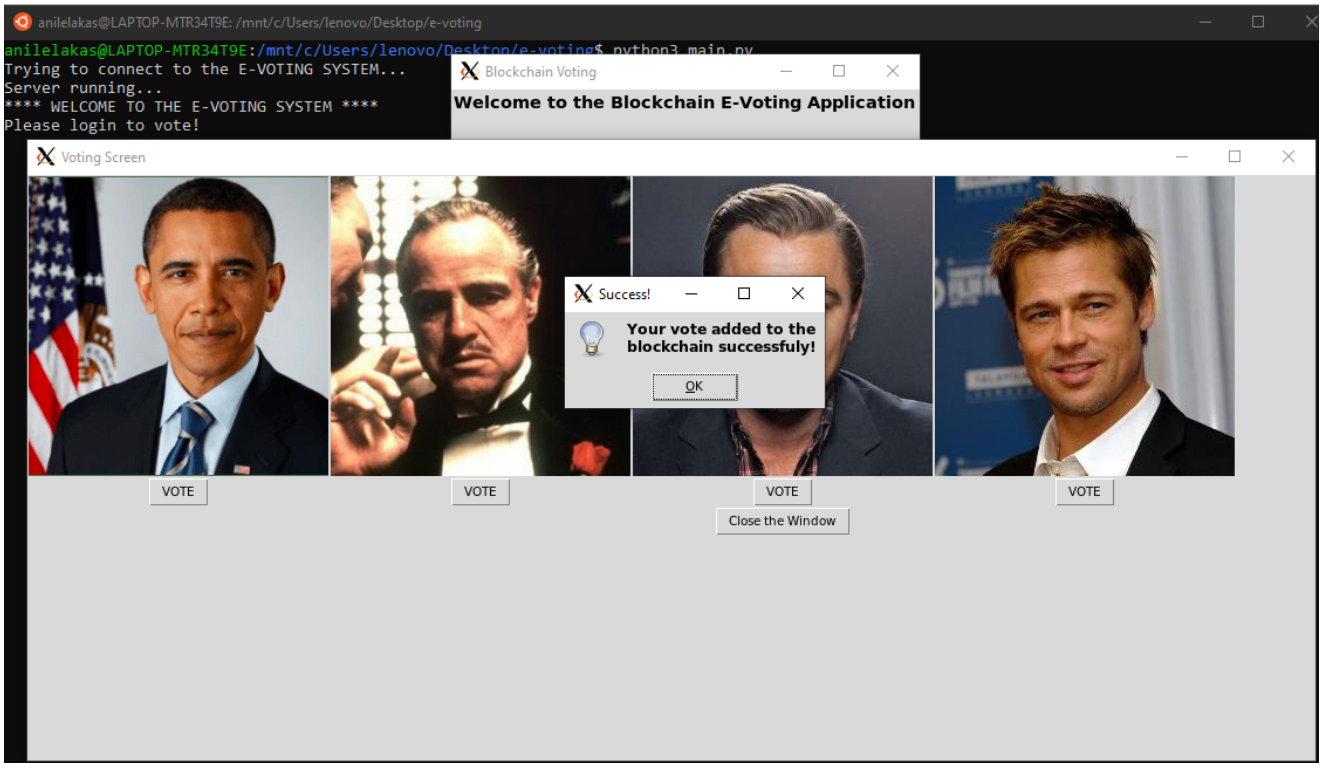

Figure 9 - Voting Screen - Not Successful Voting - If the same user tries to vote again.

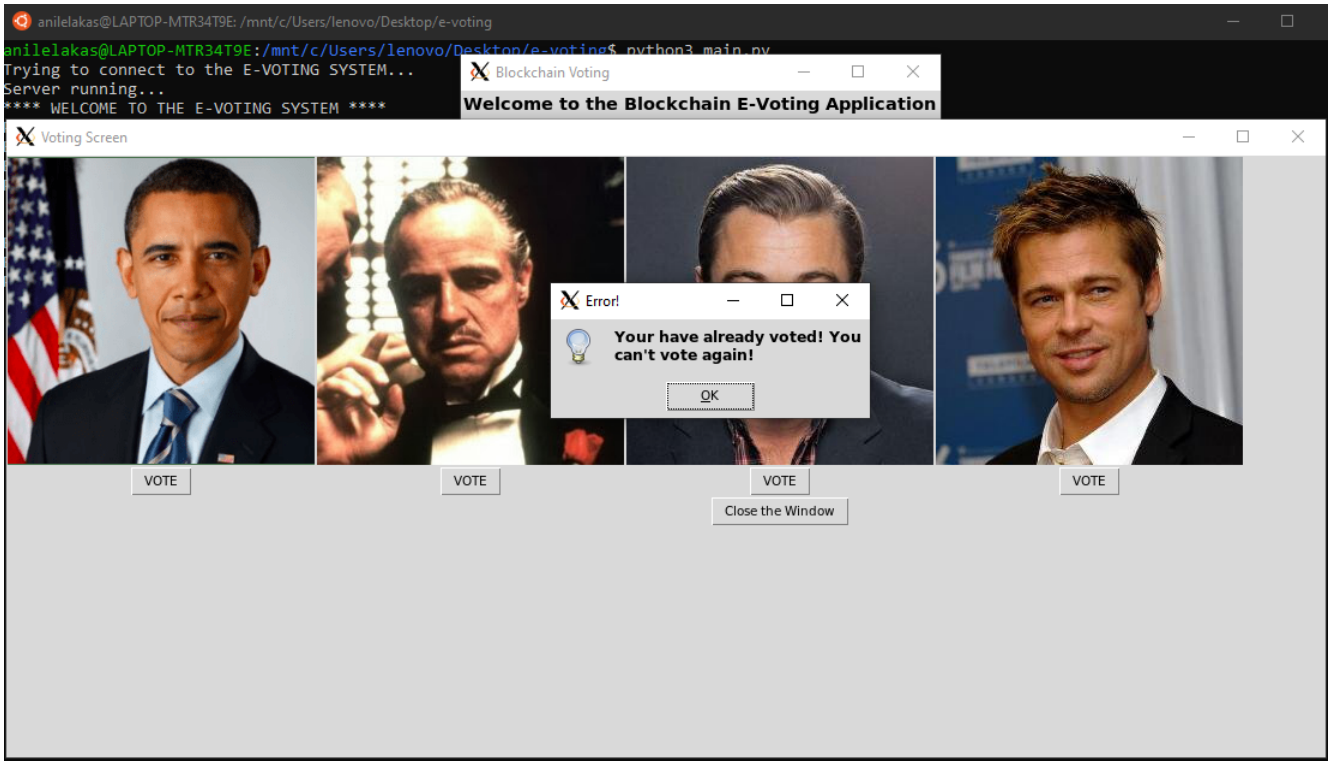

Figure 10 - Voters list - Voter's username and password tuple for login.

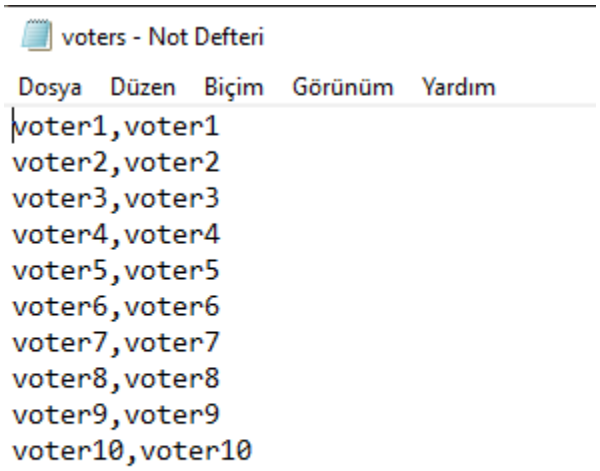

Figure 11 - blockchain.txt - "username,password,vote" data is hashed and added to the blockchain

7. References

[1] Michael Specter and J. Alex Halderman, Security Analysis of the Democracy Live Online Voting System, 30th USENIX Security Symposium (USENIX Security 21), August 2021, pages 3077--3092, USENIX Association, https://www.usenix.org/conference/usenixsecurity21/presentation/specter-security .

[2] Lifeng Guo, Qianli Wang and Wei-Chuen Yau, “Online/Offline Rewritable Blockchain with Auditable Outsourced Computation”, 04 August 2021 IEEE Transactions on Cloud Computing, DOI: 10.1109/TCC.2021.3102031 .

[3] A Study on Distributed Consensus Protocols and Algorithms: The Backbone of Blockchain Networks, DOI: 10.1109/ICCCI50826.2021.9402318

[4] Survey on Private Blockchain Consensus Algorithms, DOI: 10.1109/ICIICT1.2019.8741353

[5] An Analysis on Blockchain Consensus Protocols for Fault Tolerance, DOI: 10.1109/INCET51464.2021.9456310

[6] Changqiang Zhang, Cangshuai Wu, and Xinyi Wang. 2020. Overview of Blockchain Consensus Mechanism. In Proceedings of the 2020 2nd International Conference on Big Data

Engineering (BDE 2020). Association for Computing Machinery, New York, NY, USA, 7–12. DOI:10.1145/3404512.3404522

[7] Amazon Web Services, https://aws.amazon.com/tr/blockchain/decentralization-in-blockchain/

[8] A Survey of Distributed Consensus Protocols for Blockchain Networks, 1904.04098.pdf (arxiv.org)

[9] Howard, H. and Mortier, R., “Paxos vs Raft: Have we reached consensus on distributed consensus?”,2020, arXiv:2004.05074, doi:10.1145/3380787.3393681, Paxos vs Raft: Have we reached consensus on distributed consensus? (arxiv.org)

[10] Web3, About | Web3 Foundation

[11] Rajkumar Buyya, Christian Vecchiola, S. Thamarai Selvi, “Mastering Cloud Computing”, Morgan Kaufmann, 2013, ISBN 9780124114548, https://doi.org/10.1016/B978-0-12-411454-8.00012-7.

[12] Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008, https://bitcoin.org/bitcoin.pdf

[13] Private blockchain or database, https://medium.com/blockchain-review/private-blockchain-or-database-whats-the-difference-523e7d42edc