Advanced Journal of Science, Technology and Engineering
Volume 1, Issue 1, 2022 (pp. 52-90)
www.abjournals.org

BLOCKCHAIN-BASED CONVOLUTIONAL NEURAL NETWORK E-VOTING SCHEME

Edison Kagona

Department of Computer Science and Information Technology, International University of East Africa, Kampala, Uganda

E-mail: edisonmat15@gmail.com

Cite this article:

Edison K. (2022), Blockchain-Based Convolutional Neural Network E-Voting Scheme. Advanced Journal of Science, Technology and Engineering 1(1), 52-90. DOI: 10.52589/AJSTE-X4ASIQPQ.

Manuscript History

Received: 13 April 2022

Accepted: 5 May 2022

Published: 5 June 2022

ABSTRACT: *SQL injection attack is now the most common server-side attack in web applications whereby malicious codes are injected into the database through user input fields by unauthorized users, and this could lead to data loss or in the worst case, to database hijacking. The utilization of blockchain technology in e-voting applications is not a new thing. Many systems have been proposed using cryptography and other security techniques. In those systems, minimal involvement of the third party is observed; a problem of coercion resistance and transparency maintenance at the same time is observed and most processes have not been implemented to evaluate the systems further. This paper applies the cryptographic signatures to validate the origin and integrity of the votes by preserving the voters' choices during the election process. Then, the convolutional neural network (CNN), a regularized version of the multilayer perceptron in the system, is also applied to analyze visual imagery upon registration. Furthermore, authors provide several possible extensions and improvements that can be made as an addition to the scope of this research.*

KEYWORDS: Convolutional neural network, Blockchain, SQL injection, Decentralization, Multilayer Perceptron, Cryptographic signatures, Data Structures.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

INTRODUCTION

Worldwide, voting is a primary right and plays a significant role in constructing a democratic society. It gives individuals in a community the opportunity to voice their opinions (Joseph Siegle, 2021). Therefore, it is imperative to keep our electoral system efficient, safe, and accessible for all citizens casting their votes. Electronic voting (e-voting) refers to electronic voting machines, voting via the internet from one's computer, cell phone, or any digital device. An Electronic Voting Machine (EVM) is a simple electronic device used to record votes in place of ballot papers and boxes which were used earlier in the conventional voting systems (Kumar & Begum, 2012). Although advocates for electronic voting argue that these systems minimize costs, increase participation, and provide convenience, there are several setbacks. These include skepticism of privacy, lack of transparency, policy mitigation, fraud, and a digital divide (Ben Goldsmith, 2013).

Politicians or administrators may perhaps expect that a paper version of a certain service or process can simply be taken and put on the internet. Unfortunately, the reality is more complex, and nowhere more so than with e-voting. In every democracy, the security of an election is a matter of national authority. Recognizing that the establishment and strengthening of democratic processes and institutions is the common responsibility of governments, the electorate, and organized political forces, that periodic and genuine elections are a necessary and indispensable element of sustained efforts to protect the rights and interests of the governed and that, as a matter of practical experience, the right of everyone to take part in the government of his or her country are crucial factors in the effective enjoyment by all of the human rights and fundamental freedoms (Fogg, 2002).

The voting systems that have been utilized in most countries to authorize people to cast their ballots are either paper-based (conventional) or electronic-based (Germann, 2020). As an electronic voting system mainly relies on the internet platform, the fundamental challenge for e-voting is the significant security risks it might cause. The security challenges of electronic voting are not all centered around online voting systems, though it is well established that electronic voting machines that are often deployed in polling stations can also be hacked. For example, in September 2017 the Hacking Conference DefCon published a report titled "Report on Cyber Vulnerabilities in U.S. Election Equipment, Databases, and Infrastructure" (Lee, 2020). The report details the vulnerabilities found in several voting machines, such as compromising an AVS WinVote remotely over Wi-Fi using a vulnerability from 2003. Interestingly though, they also managed to extract 650,000 voter records from Shelby County from an Express-Poll device, which had not been correctly decommissioned. Mueller report hearsays that "By at least the summer of 2016, GRU officers sought access to state and local computer networks by exploiting known software vulnerabilities on websites of state and local governmental entities. GRU officers, for example, targeted state and local databases of registered voters using a technique known as 'SQL injection'. In one instance, in approximately June 2016, the GRU compromised the computer network of the Illinois State Board of Elections by exploiting a vulnerability in the SBOE's website. The GRU then gained access to a database containing information on millions of registered Illinois voters and extracted data related to thousands of U.S. voters before the malicious activity was identified*"* (Robert S. Mueller, 2020).

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

To moderate risks, in the past 40 years, various procedures related to the ballot-privacy, individual verifiability, eligibility, completeness, fairness, uniqueness, robustness, universal verifiability, and receipt-freeness have been widely proposed (zur Erlangung des Grades, 2010). Besides, the published protocols have implemented a variety of technologies such as blind signature, homomorphism encryption, Mix-Net, zero-knowledge proof (Wenlei Qu, 2020). Back in 2005, Estonia became the first nation to hold a legally binding general election over the internet, something that has become a norm now, while most countries are still only contemplating the option (Alexander H. Trechsel, 2020). Other countries like Australia, Canada, France, India, Mexico, Armenia, Panama, Switzerland, and the United States that have tested remote online voting, have mostly done so by introducing individual voting machines, which use vendor-produced software and are more prone to errors (Meredith Applegate, 2020).

The discussion is no longer theoretical in Africa where an increasing number of countries are turning to electronic voting or including a digital component in the voting process, such as the biometric voter recognition kits and electronic results transmission system deployed in countries. For example, from the last two elections in 2012 and 2016, Ghana had a strong digital component while Namibia held the continent's first-ever completely digital election or "e-vote" in 2014; Kenya in 2013 set up the computer system to verify voters and remit results to the national tally center in Nairobi (Wolf, 2017). Governments in many countries are eager to establish electronic voting for a variety of reasons such as convenience, reduced costs, and hope for an increase in turnout, especially amongst young people (Hall, 2021). With the rising use of the internet, SQL injection attack is the easiest method to attack e-voting systems in which attackers inject some SQL codes into the original code in the database to get sensitive information or to destroy the information. Different techniques and methods have been developed and used to protect the database. But still, attackers use this method very often because they are finding it easy to type a few deformed SQL commands into the front-end as well as back-end application. Types of SQL injection are tautologies, illegal or logically incorrect queries, union queries, piggy-backed queries, blind injection, timing attacks (Sonam Panda, 2013). Attackers inject codes using tautology statements into the authentication phase to enter into the database, which says 1=1 is always true and so the injected query becomes true even if the wrong username and password are being entered. Similarly, they use logically incorrect queries to get an error message and this message works as a hint for them to find out some information (Somayeh Dolatnezhad, 2019). Single quotes, double quotes, and backslashes are generally used in queries to make these incorrect codes work correctly. Union operator is used while injecting codes to join the injected query to the original query. Piggy-backed queries are those which use semicolons with injected codes to make duplicate codes work along with original ones. Hackers use blind SQL injection attacks by asking some true or false questions if error messages are costumed by programmers. To make a delay in operation, attackers use timing attacks and by taking advantage of this, they hack the username and password with the use of benchmarks (Vadakkanmarveettil, 2021).

Cryptography is an absolutely necessary field that ensures the security of databases. By applying the encryption method, database attacks can be prevented. Today, cryptography has been employed to offer several benefits to electronic voting and counting solutions (Masood Ahmad, 2020). It is the science of protecting information by transforming it into a secure format. Besides, cryptography also covers the obfuscation of information in images using techniques such as microdots or merging. Ancient Egyptians were known to use these methods

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

in complex hieroglyphics, and Roman Emperor Julius Caesar is credited with using one of the first modern ciphers (Mirev, 2021). When transmitting electronic data, the most common use of cryptography is to encrypt and decrypt email and other plain-text messages. The simplest method uses the symmetric or "secret key" system. Here, data is encrypted using a secret key, and then both the encoded message and the secret key are sent to the recipient for decryption (Peter Loshin, 2021). If the message is intercepted, a third party has everything they need to decrypt and read the message. To address this issue, cryptologists devised the asymmetric or "public key" system. In this case, every user has two keys: one public and one private. Senders request the public key of their intended recipient, encrypt the message and send it along. When the message arrives, only the recipient's private key will decode it, meaning theft is of no use without the corresponding private key (Shinder, 2008). On 13th September 2018, computer scientist J. Alex Halderman rolled an electronic voting machine onto a Massachusetts Institute of Technology stage and demonstrated how simple it is to hack an election. In a mock contest between George Washington and Benedict Arnold, three volunteers each voted for Washington. But Halderman, whose research involves testing the security of election systems, had tampered with the ballot programming, infecting the machine's memory card with malicious software. When he printed out the results, the receipt showed Arnold had won 2 to 1. Without a paper trail of each vote, neither the voters nor a human auditor could check for discrepancies. In real elections too, about 20 percent of voters nationally still cast electronic ballots only (Schwartz, 2018).

The utilization of blockchain technology in e-voting applications is not a new thing. Many systems have been proposed using cryptography and other security techniques. In such cases, minimal involvement of third party is observed and a problem of coercion resistance and transparency maintenance at the same time is observed. However, most of those processes have not been implemented to evaluate the systems further. In the past years, securing data on the electronic voting systems, the blind signature theorem was used to make sure there were no links between voters and ballots (Chaum, 2019). Since then, many scholars have continued to show interest in the subject and a lot of research has been done. For example, Yi Liu and Qi Wang proposed a decentralized e-voting protocol without the existence of a trusted third party. The protocol was designed but required further optimization and implementation according to the specified conclusion; the balancing of transparency and coercion-resistance was a possible future work (Wang, 2017). Antony Lewis et al. (2018) described blockchain as an open, distributed ledger of historical records that uses cryptography and digital signatures. In his paper he also mentioned the logic of blockchain and how it works. On explaining the aftermath of resolving conflicts, he introduced an idea of not broadcasting a block intentionally. Two blocks are created, and one can be left as being not broadcasted, the un-broadcasted block is broadcasted when desired (Lewis, 2018). Clement Chan Zheng Wei and Chuah Chai Wen (2020) proposed a blockchain-based electronic voting protocol. This protocol was implemented using blockchain to turn election protocol into an automated control system without relying on any single point of entity (Wei, 2020). However, this proposed protocol does not give a step-by-step description of the solution and does not include any proposed algorithm or encryption standard used in registration, client side or server-side security of the voting process. Another proposed e-voting procedure is a blockchain based secured e-voting by using the assistance of smart contract. This protocol utilizes smart contract into the e-voting system to deal with security issues, accuracy and voters' privacy during the vote. The protocol results in a transparent, non-editable and independently verifiable procedure that discards all the intended fraudulent activities occurring during the election process by removing the least

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

participation of the third party and enabling voters' right during the election (Kazi Sadia, 2020). However, the proposed solution does not show how the voters' data is securely captured and verified using an encryption standard or hashing algorithms.

The value of blockchain technology and convolutional neural networks (CNNs) is now dawning on developers focusing on the accuracy of data results. This has led to a new race of developing new applications of this technology of which one is the application of blockchain technology and artificial neural networks in elections (Zhonghua Zhang, 2021). A convolutional neural network consists of an input layer, hidden layers, and an output layer. In any feed-forward neural network, the middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution (Brownlee, 2018). In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix (Ricardo Ñanculef, 2020). This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers (Mishra, 2020).

The combination of CNNs with blockchain has been introduced in many papers like "Agent Architecture of an Intelligent Medical System Based on Federated Learning and Blockchain Technology" where the authors propose building an agent with a consortium mechanism for classification results from many machine learning solutions. The main advantages of that proposed system are based on the implementation of blockchain technology elements and threaded federated learning (Dawid Połap, 2021). The second paper is the "Blockchain Technology and Neural Networks for the Internet of Medical Things." In this paper, the author proposed a federated learning approach that uses decentralized learning with blockchain-based security and proposition that accompanies the training of the intelligent systems using distributed and locally-stored data for the use of all patients. In all the previously proposed papers, the authors have targeted different sectors proposing solutions that fuse blockchain into the machine learning methods. The proposed solutions in the area of e-voting have only employed blockchain technology which has not yet produced satisfactory results as studied in the above literature. Therefore, to solve some of the weaknesses of the proposed existing systems, this paper applies the cryptographic signatures to validate the origin and integrity of the votes by preserving the voters' choices during the election process. Then, the convolutional neural network (CNN), a regularized version of the multilayer perceptron in the system, is also applied to analyze visual imagery upon registration. The hypothesis of this paper is that blockchain establishes a system of creating a distribution agreement in the digital online world. This allows participating entities to know for certain that a digital event happened by creating a convincing record in a public ledger. It opens the door for developing a democratic open and scalable digital economy from a centralized one that is susceptible to server-side attacks. It is possible to as well integrate sophisticated analytic models that personalize social and e-mail messaging using data generated by social media activity, hence guiding in strategic decision making.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

METHODOLOGY

User Registration Process Using a Facial Recognition Module

For a voter to be legible to vote, he or she has to be registered on the Electoral Authority System (EA). The registration process involves the use of a biometric digital device. This device is a security identification and authentication device. It uses automated methods of recognizing and verifying the identity of a living person based on a physiological or behavioral characteristic. These characteristics include fingerprints, facial images, iris, and voice recognition. In the proposed system, we focus on facial recognition. Below is a high-level overview of the real-time face recognition process used in the registration process of the users on the proposed system.

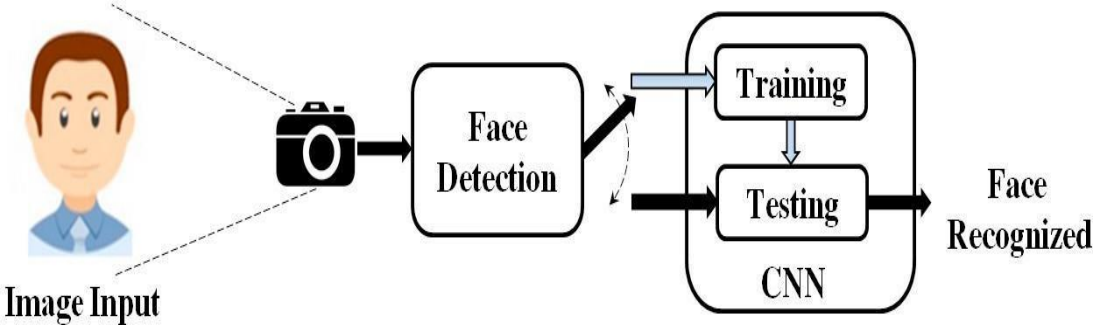


Figure 1: Illustration of the real time facial recognition module

The data collected is pushed to the CNN architecture. This architecture consists of repetitions of a stack of several convolution layers and a pooling layer, followed by one or more fully connected layers.

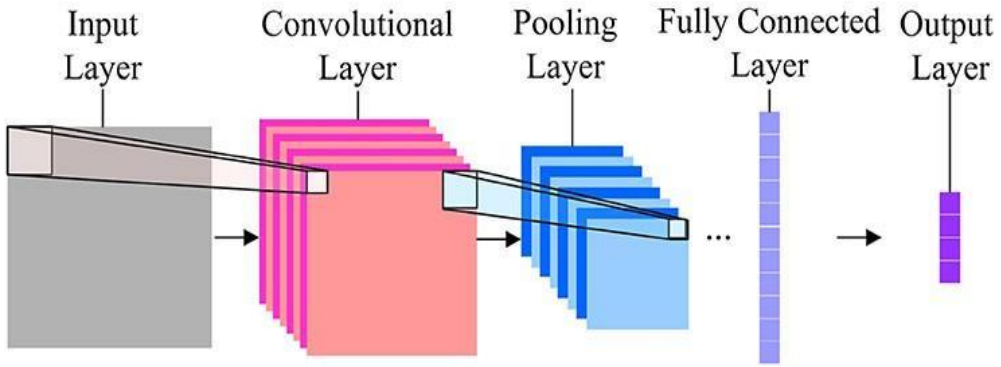


Figure 2: CNN architecture illustrating the stack of several convolution layers, a pooling layer, and fully connected layers

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

The proposed system uses forward propagation to transform the input data into the desired output.

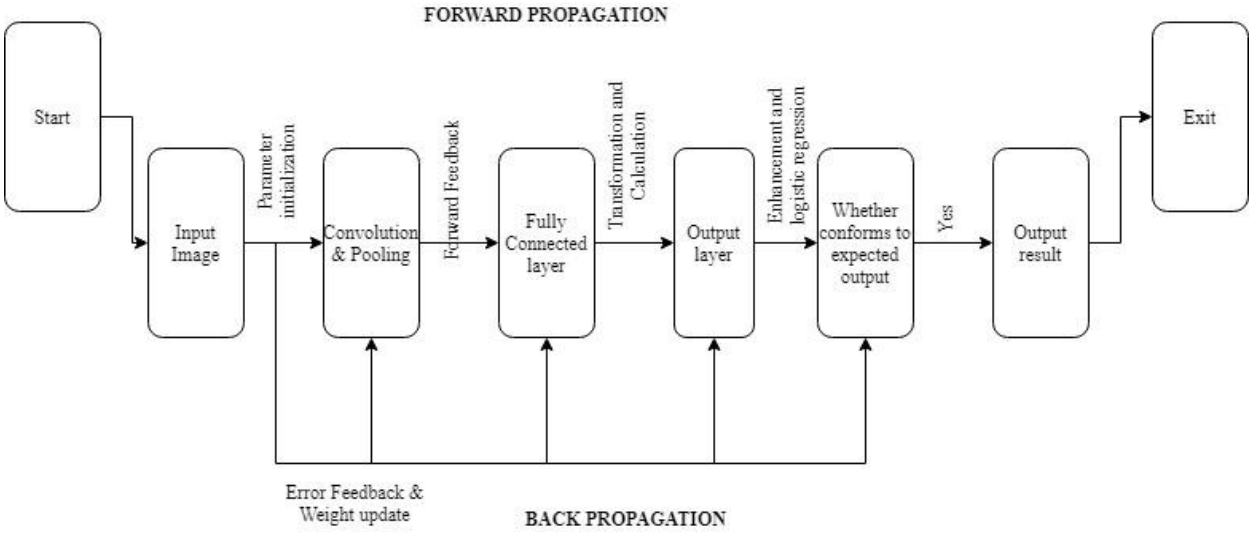


Figure 3: Illustration of foward propagation and backpropagation processes

a) Convolutional Layers

The layer's parameters consist of a set of learnable filters (or kernels) of a fixed size which slides in a window fashion to perform the convolution operation on the windowed image to extract features. Padding is applied to the size of the input image to overcome uneven mapping with filter size. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the filter entries and the input, producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input. The input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) x (feature map height) x (feature map width) x (feature map channels). Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

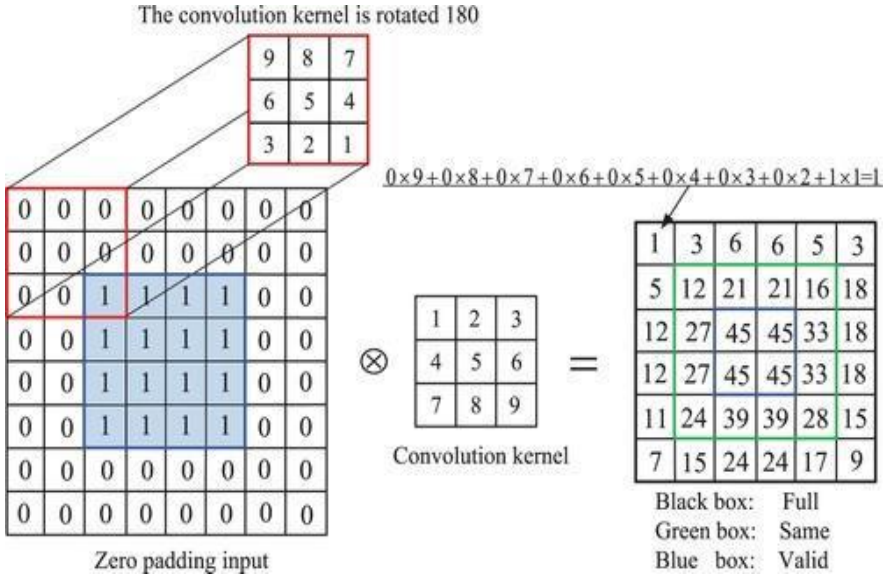


Figure 4: Illustration of the convolution operation

b) Pooling Layer

While pooling layers contribute to local translation invariance, they do not provide global translation invariance in a CNN, unless a form of global pooling is used. The pooling layer commonly operates independently on every depth, or slice, of the input and resizes it spatially. A very common form of max-pooling is a layer with filters of size 2×2 applied with a stride of 2, which subsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations:

- Max pool with a $2x2$ filter and stride 2
- Max pool with a $2x2$ filter and stride 2

For instance, mean-pooling can extract the average value of the feature points and has the effect of maintaining the relative background, while max-pooling can extract the maximum value of the feature points and achieve better texture extraction. Specifically, for the mean-pooling, if a feature map with a size of $4X4$ is sampled by using a kernel with a size of $2X2$, the output is a feature map with a size of $2X2$.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

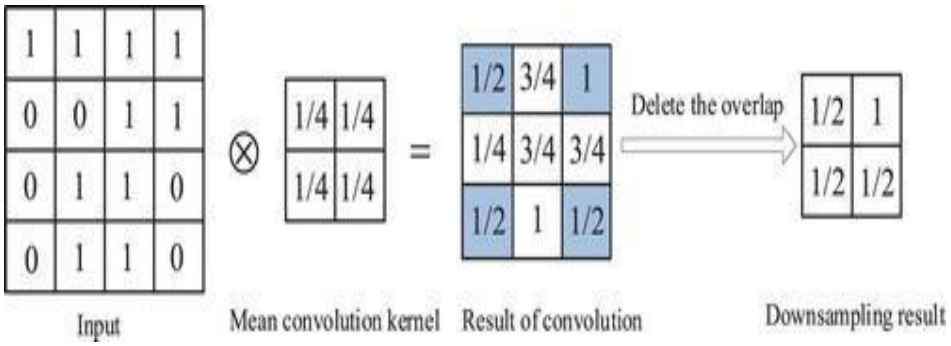


Figure 5: Illustration of the pooling operation

c) ReLU Layer

The rectified linear unit (ReLU) is used to apply the non-saturating activation function $f(x) = max(0, x)$ at some level to effectively remove negative values from an activation map by setting them to zero. It introduces nonlinearities to the decision function and in the overall network without affecting the receptive fields of the convolution layers.

d) Fully Connected Layer

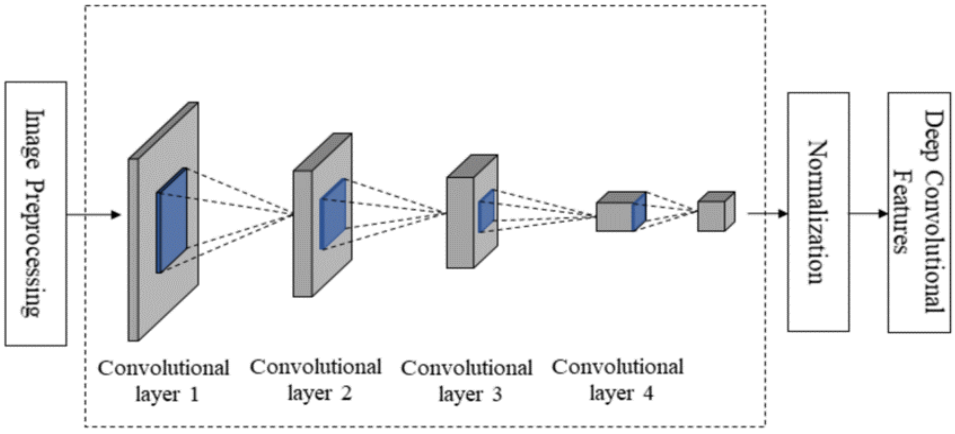


Figure 6: Illustration of image processing through the fully connected layers

After several convolutional and max-pooling layers, the final classification is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer. Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset (vector addition of a learned or fixed bias term).

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

Message Transmission Procedure to the Blockchain System

Introducing message transmission on the blockchain. After the registration process using a facial recognition module, every user is associated with an asymmetric key pair, i.e., a public key and a private key. As the abstract message structure depicts below, senders fill out the area with their public keys, receivers' public keys, and message contents. Afterward, senders use their private keys to sign messages and send them to the blockchain P2P network. In this setting, messages are collected and packed into a block during a period. As a message spreads over the network and is recorded on the blockchain, it is obtained from the blockchain by the receivers.

Table 1: Example of the message structure

Message	**Structure**
Sender	pk_{Voter}
Receiver	pk_{EA}
Message	*Message content*

System Notation and Definitions

Table 2: Illustration of the notation structure

j	$Voterj$
$jpub$	The public key of j. uniquely identifies j, also serves as signature verifying key.
$jpriv$	The private key counterpart of j_{pub} is used as a signing key.
$EAsign(m)$	Digital signature produced by EA over message m.
CHj	Voters' choice in the election.
$DCHj$	Digital commitment for CH_j
OVj	DCH_j opening value. CH_j cannot be derived from DCH_j without the opening value.
x/y	Inclusion of values x and y in a single message.
$Etokenj$	Eligibility token.

1. **Voter**: A voter, identified by one's public key, $jpub$, is considered an object that is permitted to cast a vote towards one of the candidates. Voters can access the electronic voting platform through a voting client-side application in any browser of choice and preference, the security of which is assumed for example the tor-browsers. During elections, using CHj, a voter makes a choice that ranges from a set of predefined choices. In order to assure fairness, the voter reveals only a digital commitment over the said choice. $DCHj$ only reveals the choice itself only during the counting stage of the election.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

2. **Electoral Authority *(EA)*:** In order for the e-voting system to provide a declaration that only eligible voters are able to vote, it was deemed necessary for an Electoral Authority to be introduced. This is any voting officer assigned the role to monitor the voting process to do verification and authorization of the eligible voters generated from the facial recognition module. For a user to be judged eligible, one must authenticate oneself to the Electoral Authority *(EA)*, and receive a token that proves one's eligibility to vote. The eligibility token $Etokenj$ takes the form of a digital signature over a voter $jpriv$ and $DCHj$.

   $$E_{tokenj} = EA_{sign}\ (j_{pub} \mid DCH_j).$$

   For the *EA* to judge whether voters are eligible or not, it is required for it to maintain a list of all the voters that are allowed to participate in the elections. The *EA* is also required to have access to voter authenticating information in order to have the ability to authenticate eligible users.

3. **Vote:** A vote is a message of predefined structure that is the equivalent of a bitcoin transaction. A vote is required to include a ballot and any other information that the practical implementation of the system requires it to include. Each vote x, once included in the blockchain, is identified by a vote id $V(idx)$, a value that uniquely recognizes a vote.

4. **Ballot:** A ballot, Bj is the digital representation of the physical ballot, i.e., the paper where the choice of a voter is written on. A ballot is considered sealed when the opening value of the digital commitment has not been revealed and thus no party, other than the voter, can determine the way a voter voted. Once the opening value has been revealed and the choice of the voter is publicly known, the ballot is considered open. The public key included in the ballot symbolizes the owner of the ballot and by extension the owner of the vote.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

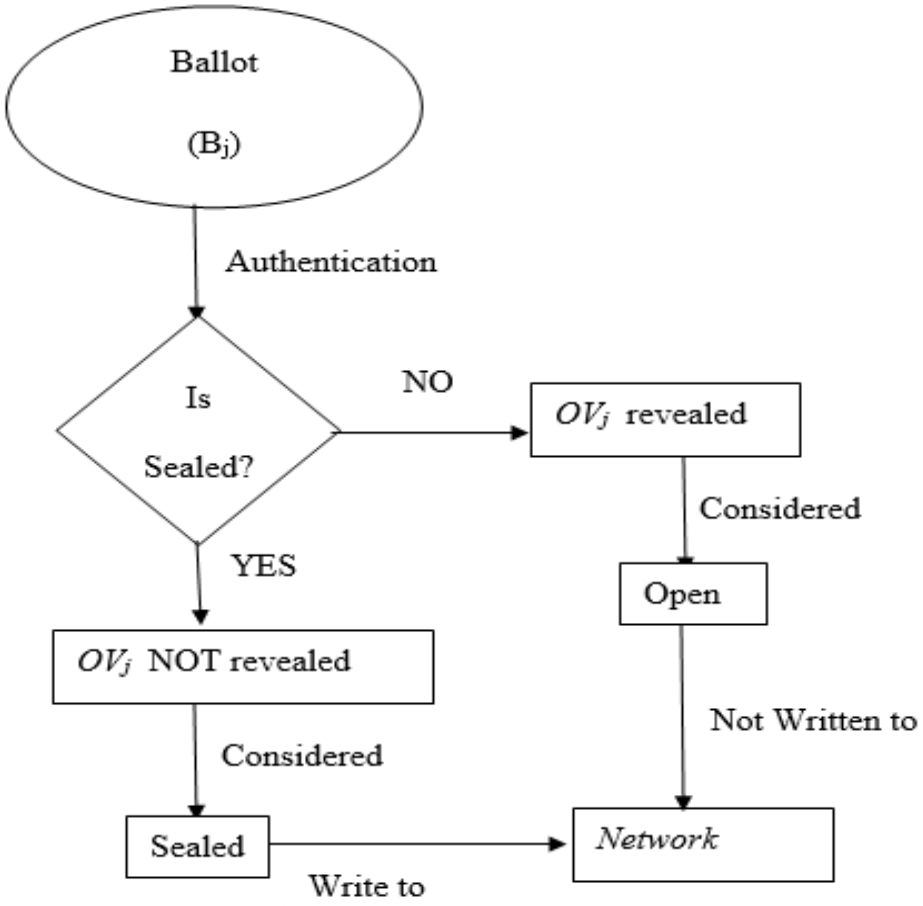


Figure 7: Data flow diagram illustrating the ballot authentication procedure

System Execution in Phases

a) The Initialization Phase:

In this phase, rules governing the elections are predetermined by the electoral authority *(EA)*. The Ethereum blockchain and all other modules of the system are initialized. The voting office under *EA* decides the duration of the voting process, the phases involved, and whether vote cancelation will be granted or not. This communication is published and communicated through their official website and using other communication media houses. The blockchain infrastructure and the *EA* are collectively governed by the same rules. The *EA* during the initialization phase will be provided the list of the voters that are eligible to vote as well as a way to authenticate those users. A pair of signing and verifying keys for the public signature scheme will be generated and the verifying key will be publicized as a system-wide parameter. The blockchain will be initialized with an initialization block that will serve as the genesis block of the chain. The initialization block does not contain any votes, but instead, it contains all the information of the election, including the *EA's* signature validating key, the set of valid choices the voters can choose from, and so on. This way, a blockchain is tied to a specific

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

election and all the system parameters become part of the blockchain and thus dispute over them is prevented.

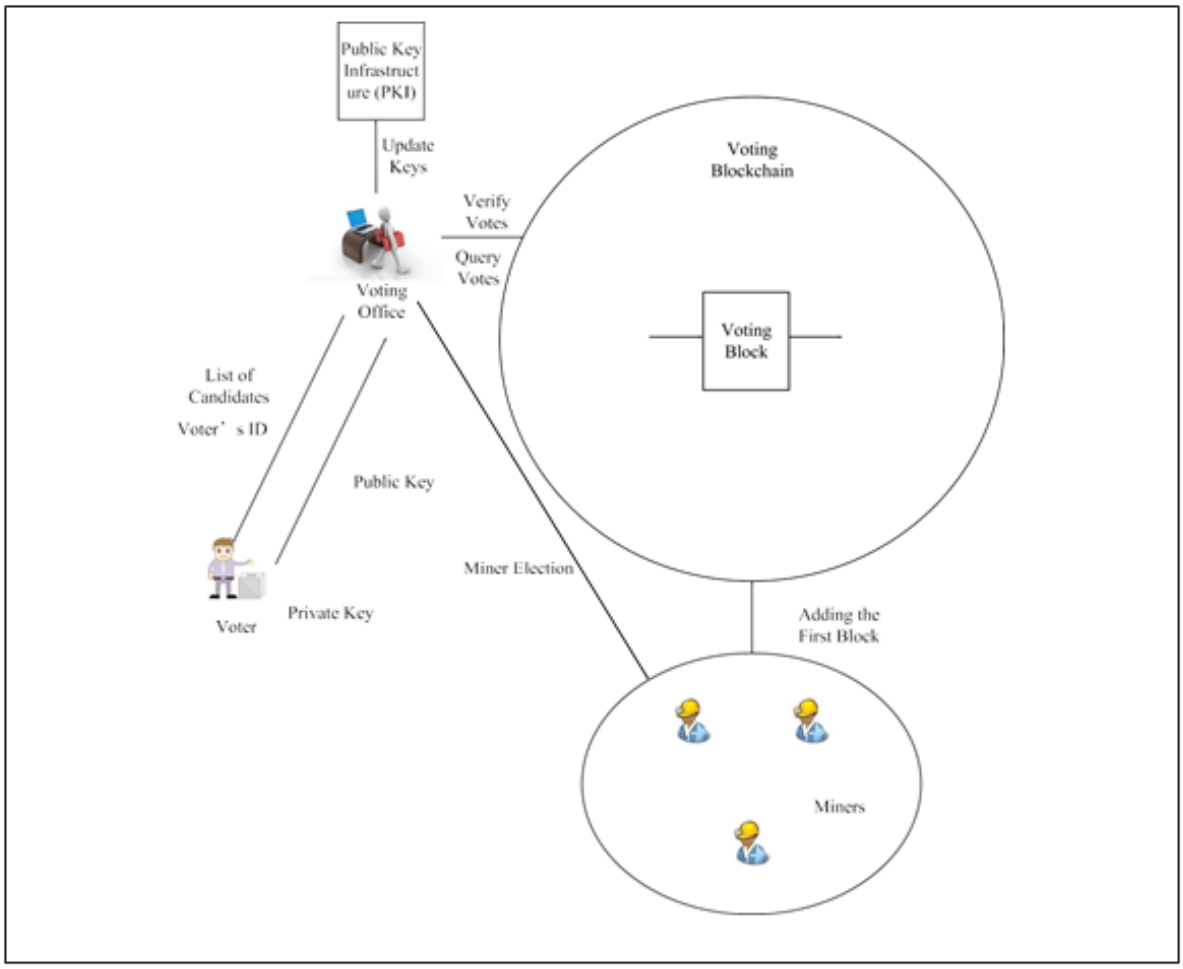


Figure 8: Initialization of the voting process to the blockchain

b) The Preparation Phase:

In this phase, the voter, using the client-side application, is called to authenticate themself to the electoral authority *(EA)*. The *EA* will use the list of eligible voters along with the authentication information acquired during the initialization phase to determine whether the aspiring voter is eligible to vote. If the voter is judged eligible, the *EA* will proceed to the following steps; otherwise, the voter j is rejected and the *EA* does not proceed with the rest of the phase. All the following information will be exchanged through an authenticated and secure channel. Once deemed eligible, the voter j client will generate a public key pair, whose public counterpart $jpub$, will be used as a pseudonymous identity of the voter and will also serve as one's verifying key. It will also prompt the voter to make one's choice, CHj of the

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

predetermined choices that will be accepted by the system and a digital commitment scheme will be used in order to generate $DCHj$, the digital commitment of the voter's choice.

To prove one's eligibility, the voter needs to send both $DCHj$ and $jpub$ to the *EA* to be signed by it. To avoid the possibility of the *EA* linking a voter's true identity to one's vote, the RSA blinding signature scheme is used at this level. The client will apply a blinding function on the message, $blind(jpub|DCHj)$ and send it to the *EA* that will then sign the blinded message and send it back. Once the message is received, the client will unblind the signed message$sign-1(sign(jpub|DCHj)EA) = sign(jpub|DCHj)EA$ and will end up with a valid eligibility token $Etokenj$.

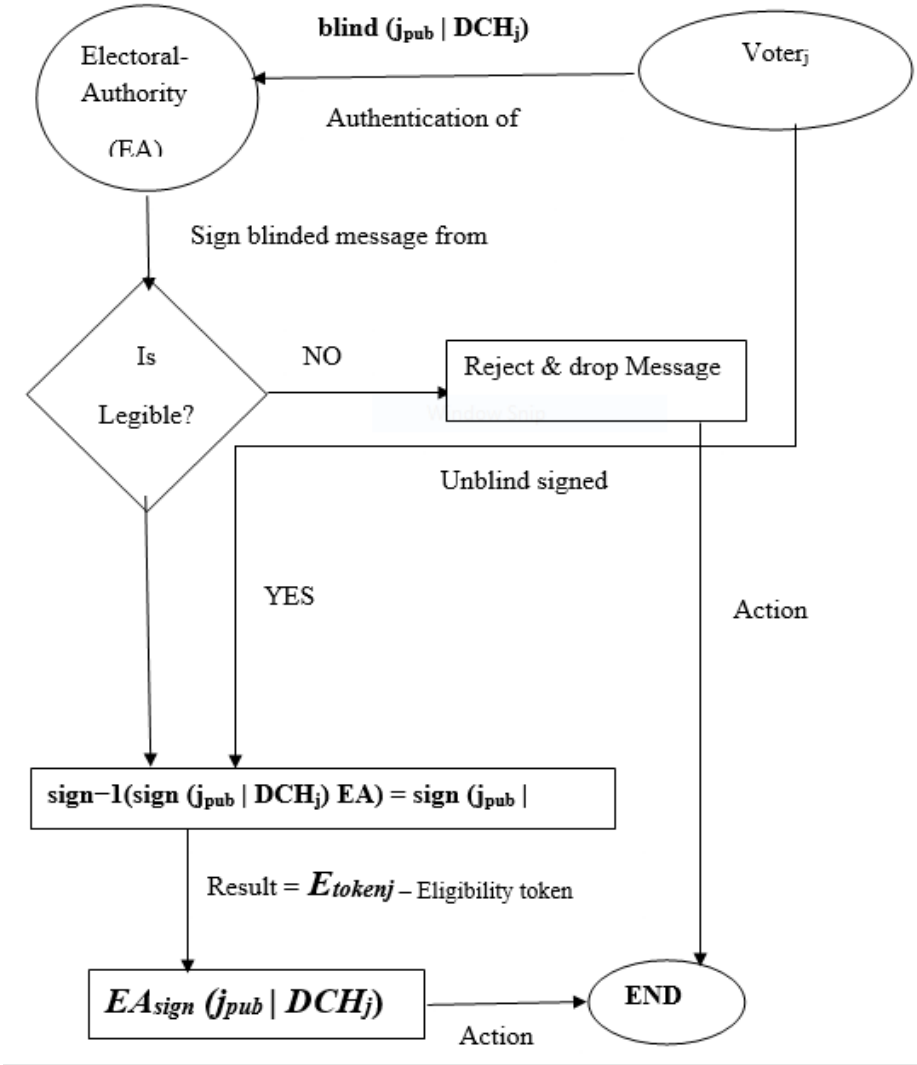


Figure 9: Data flow diagram describing the signing procedure and eligibility token generation

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

c) The Voting Phase

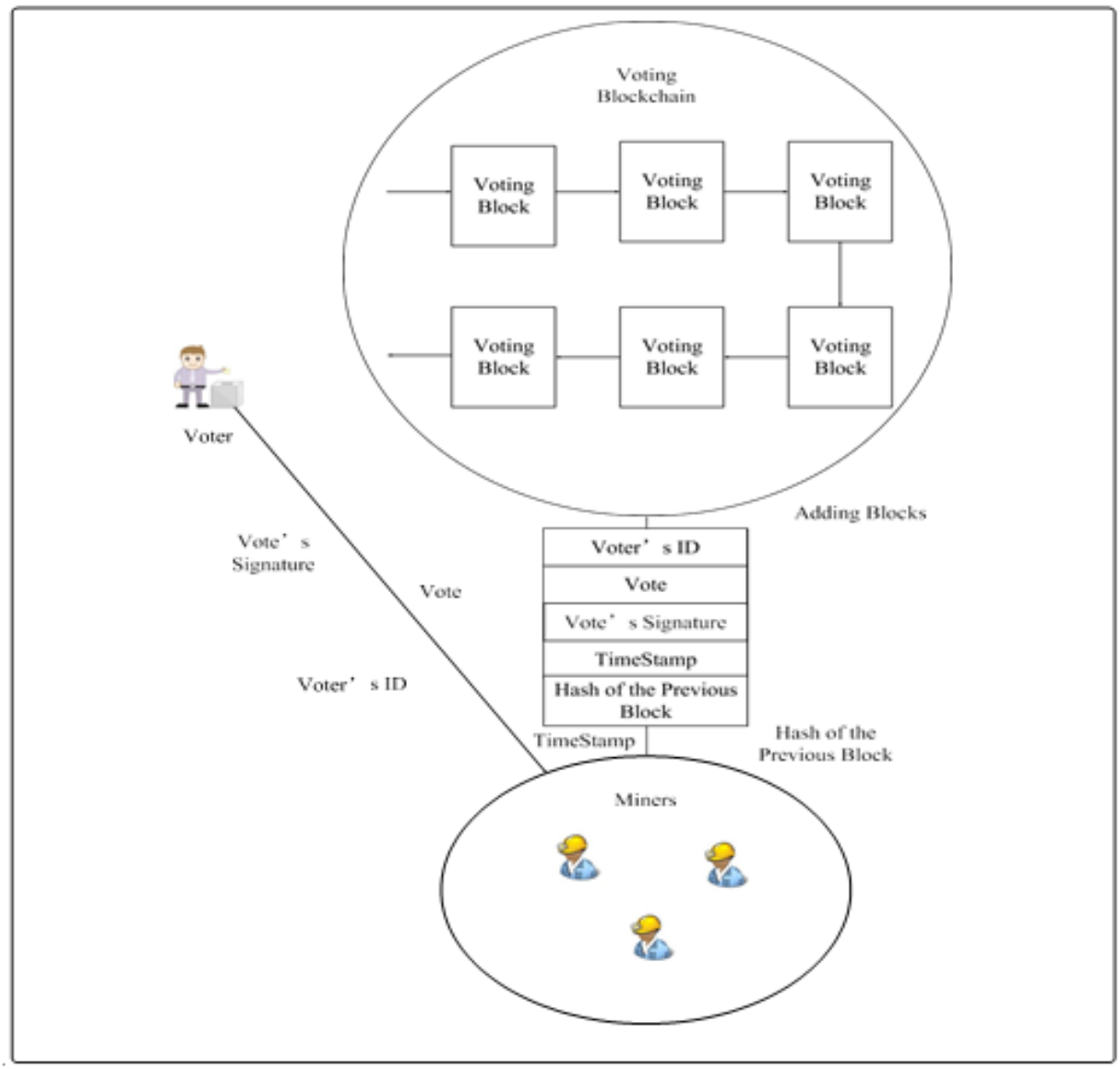


Figure 10: Illustration of the voting process on the blockchain

- A voter uses SHA-256 to generate the hash value of $H = Hash(ID + Vote + Timestamp)$.
- The voter uses his/her private key to generate a signature S of the hash value H.
- The voter sends ID, vote, timestamp, S to the miner.
- The miner obtains the public key from the PKI according to the voter's ID.
- The miner uses SHA-256 to generate the hash value of $H = Hash(ID + Vote + Timestamp)$.
- The miner uses the public key to verify S *and* get Hr.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

During the voting phase, every voter constructs and then broadcasts to the network their vote. Each miner is also responsible for collecting votes, validating them and inserting the valid ones in the blockchain. In order for the vote to be accepted as a valid one to be added into a block, the miner has to make sure that the voter has not previously cast a vote, the EA's signature included in the ballot is validated and the vote adheres to the predefined structure. If any of those checks fail, the vote is discarded as an invalid vote.

d) The Counting Phase

During the counting phase, all voters are called to reveal their final choice by broadcasting to the network a ballot opening message, OBj, containing the $V(id)$ of their final vote in the blockchain, the opening value of their vote commitment, and a signature over both values.

$$OBj = V(idx)|ovj|sign(V(idx)|ovj)EA$$

All nodes of the network are responsible for collecting the ballot opening messages and verifying that the signature validates with the public key of the owner of the vote $V(idx)$. If the signature is verified, the voters will then broadcast the messages to their adjacent peers, and proceed with including the vote in their count. All peers should reach the same result since they operate on the same blockchain.

Evaluation of the Proposed System

In this section, we examine the extent to which the system satisfies the electronic voting properties.

- **Eligibility**: For a vote to be considered cast, for it to be accepted and written on the blockchain, it needs to include a valid ballot. Each ballot needs to include a valid signature of the *EA* over $jpub$ and $DCHj$; otherwise, it is dropped by the network. The *EA* provides signatures only to authenticated voters that have been included in the list of eligible voters compiled during the initialization phase and that haven't requested to vote before. This means only eligible voters can vote and they can acquire only one eligibility token and thus cast only one valid vote. The eligibility property also breaks when an eligible voter succeeds in casting a vote more than once. This, however, is not possible since all nodes, during the voting phase, will refuse to include to the blockchain a vote that has already been cast.

- **Privacy:** The design of the system guarantees that no party can determine how a voter voted at any point during the system run. The only link between the real identity of the voter and one's vote is an individual's public key that acts as a pseudonymous identity. The only entity that would possibly expose the said link would be the *EA*, since it is the only entity that the voter would need to reveal their true identity in order to obtain proof of their eligibility, during the preparation stage. A blind signature is used to avoid any party from being able to verify how a voter voted. Blind signatures provide a way for the Electoral Authority (EA) to produce a valid signature on the digital commitment ($DCHj$) and public key of a voter without being able to determine the public key ($jpub$) or the digital commitment ($DCHj$).

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

- **Individual Verifiability**: Due to the public nature of the ledger, each voter can verify that their vote has been inserted in the blockchain, and thus has been counted. Each voter is also responsible for counting the votes and thus they can ensure that the result includes their vote.

- **Universal Verifiability**: Since the ledger is public, every voter can verify that the votes have been counted correctly, by simply counting the votes. External auditors can also verify the results by obtaining a copy of the blockchain, making sure that the votes in it are legitimate, for example, that signature is validated, duplicates don't exist, and once all checks are complete, auditors can count the votes and compare their results against the official election tally. The fact that rules governing the election are included in the genesis block of the blockchain further facilitates the election's verifiability since their integrity is guaranteed and thus disputes over them become irrelevant.

Since only voters are allowed to participate in the elections, the suggested blockchain to be used is a permissioned one. The voters can use their eligibility tokens as proof that they can participate in the blockchain. This makes the environment integrally more secure.

Prototype Implementation and Testing

Implementation

To implement a part of the system in a prototype for testing and evaluation, we used a few dependencies:

Node Package Manager (NPM). This dependence comes with Nodejs.

- **Truffle Framework:** This allowed us to build a decentralized demo on the Ethereum blockchain. It provided a suite of tools that allowed us to write smart contracts with the solidity programming language. It also enabled us to test our smart contracts and deploy them to the blockchain.

- **Ganache:** This dependency provided us with addresses on our local Ethereum blockchain. Each account was preloaded with 100 fake ether for testing purposes. Below is a screenshot of the Ganache. Each account address serves as a unique identifier to each voter in this demo of our election using the above protocol.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

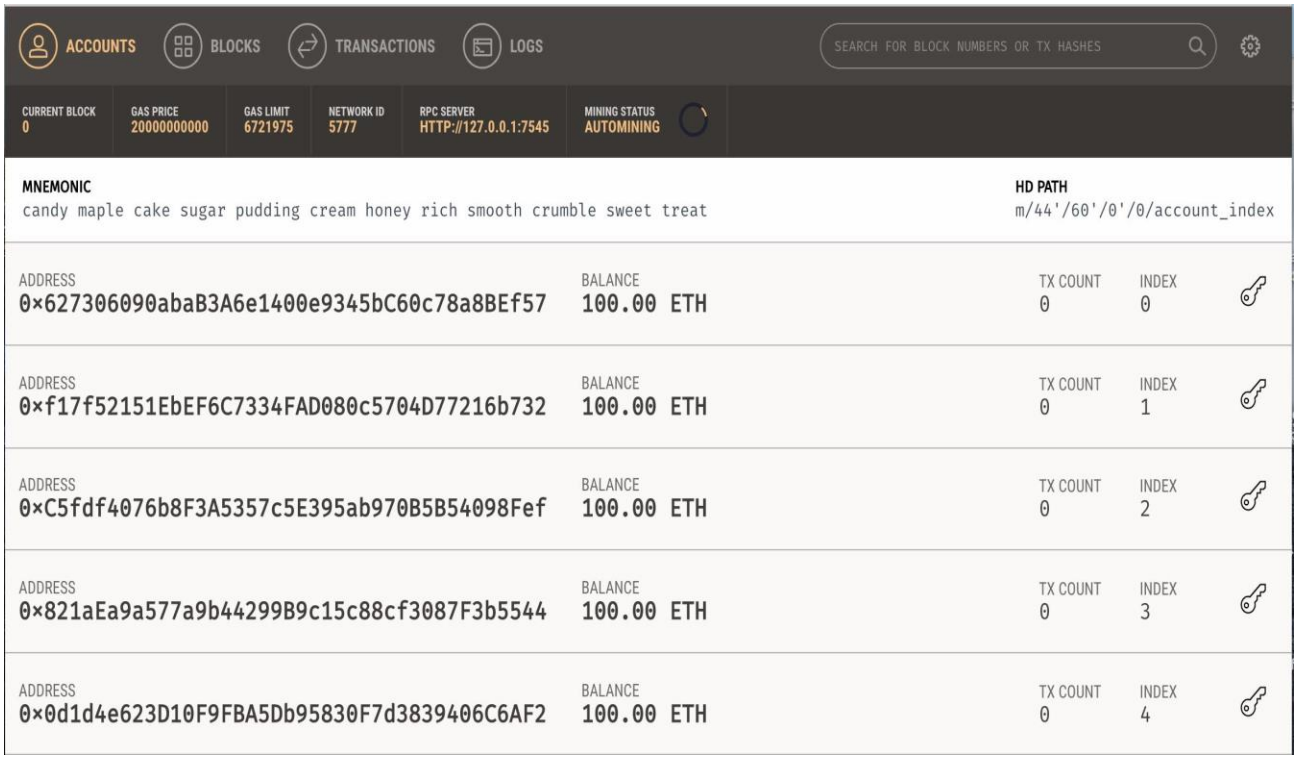


Figure 11: Ganache screenshot preloaded with 100 ether for testing purposes

- **Metamask:** To use the blockchain, we were required to connect to it since the blockchain is a network as we learned from our study. Therefore, we installed a special browser extension to use the Ethereum blockchain. Using the Metamask, we connected to our local Ethereum blockchain with our account and interacted with our smart contract.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

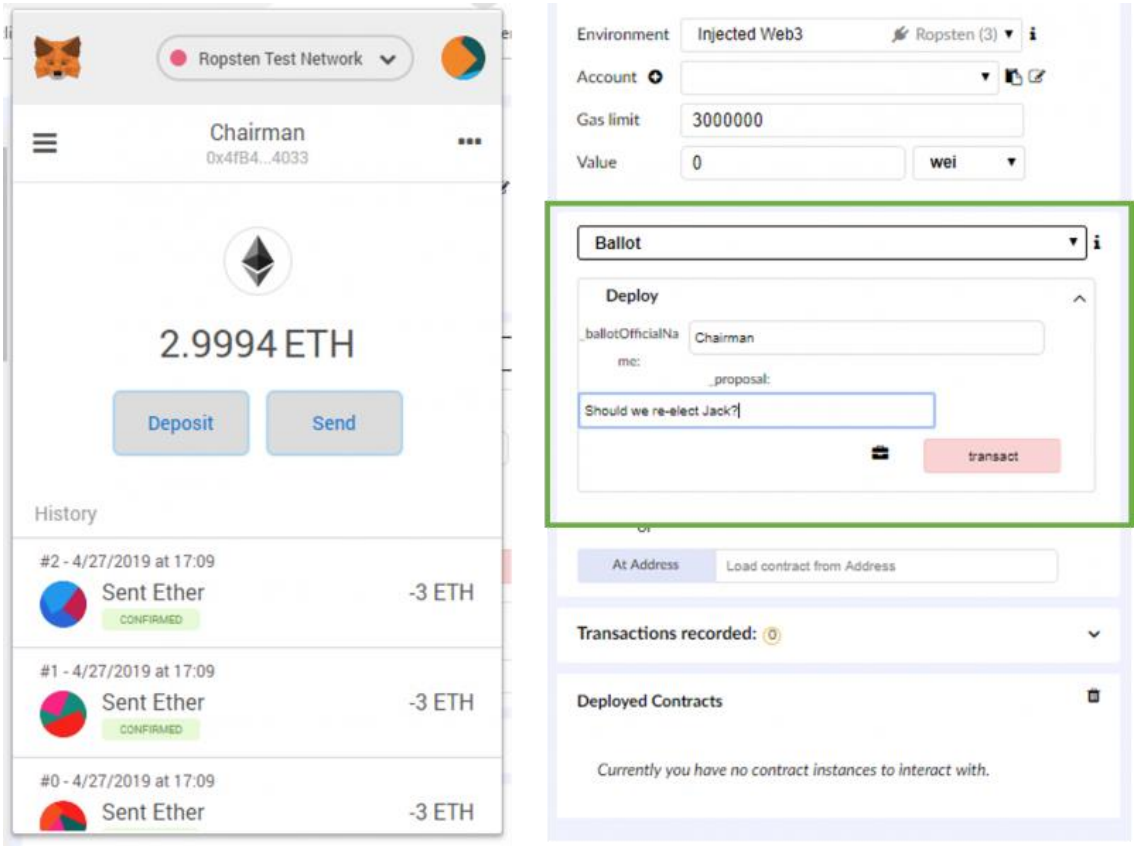


Figure 12: MetaMask screenshot

a) Signing and Verifying Messages

Signing a message with a private key does not require interacting with the Ethereum network. It can be done completely offline. In this section, we demonstrate the off-chain computation of messages signed and then verified on chain using a smart contract. The message signing procedure involves two parties, a user that wants to obtain signatures on their messages, and a signer that owns their secret signing key. At the end of the process, the user obtains the signer's signature on message (m) without the signer learning anything about the message. The signature algorithm Ethereum has built-in support for is the Elliptic Curve Digital Signature Algorithm (EDCSA). In our protocol, the sign method calculates an Ethereum specific signature with:

sign (keccak256("\x19Ethereum Signed Message:\n" + len(message) + message))).

Adding a prefix to the message makes the calculated signature recognizable as an Ethereum specific signature. This prevents misuse where a malicious decentralized application can sign arbitrary data (e.g., transaction) and use the signature to impersonate the victim.

Recovering the Message Signer in Solidity. Here we reviewed the ECDSA signature which consists of two parameters, ***r*** and ***s***. Signatures in Ethereum include a third parameter, ***v***, which provides additional information that can be used to recover which account's private key was used to sign the message. This same mechanism is how Ethereum determines which account

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

sent a given transaction. Solidity provides a built-in function ecrecover() that accepts a message along with the *r*, *s*, and *v* parameters and returns the address that was used to sign the message.

b) Signature Verification

Steps taken in the signing and verification of the message:

First, we created a message to be signed

i. We hash the message

ii. We sign the hash (off chain, keep your private key secret)

iii. Recreate the hash from the original message

iv. Recover signer from signature and hash

v. Compare recovered signer to claimed signer

In the figures below, we demonstrate the message creation, hashing of the message, signing of the hash off-chain and verification process.

```
contract VerifySignature {
    /* 1. Unlock MetaMask account
    ethereum.enable()
    */

    /* 2. Get message hash to sign
    getMessageHash(
        0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C,
        123,
        "Edison Kagona, MSc Computing, UTAMU",
        1
    )

    hash = 0xcf36ac4f97dc10d91fc2cbb20d718e94a8cbfe0f82eaedc6a4aa38946fb797cd
    */
    function getMessageHash(
        address _to, uint _NIN, string memory _message, uint _nonce
    )
        public pure returns (bytes32)
    {
        return keccak256(abi.encodePacked(_to, _NIN, _message, _nonce));
    }
```

Figure 13: Contract VerifySignature

In the above screenshot, we implemented a message with a receiver public key, the _NIN representing an ID of the sender, message body of the sender and the _nonce. The _nonce is a number or bit string used only once.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

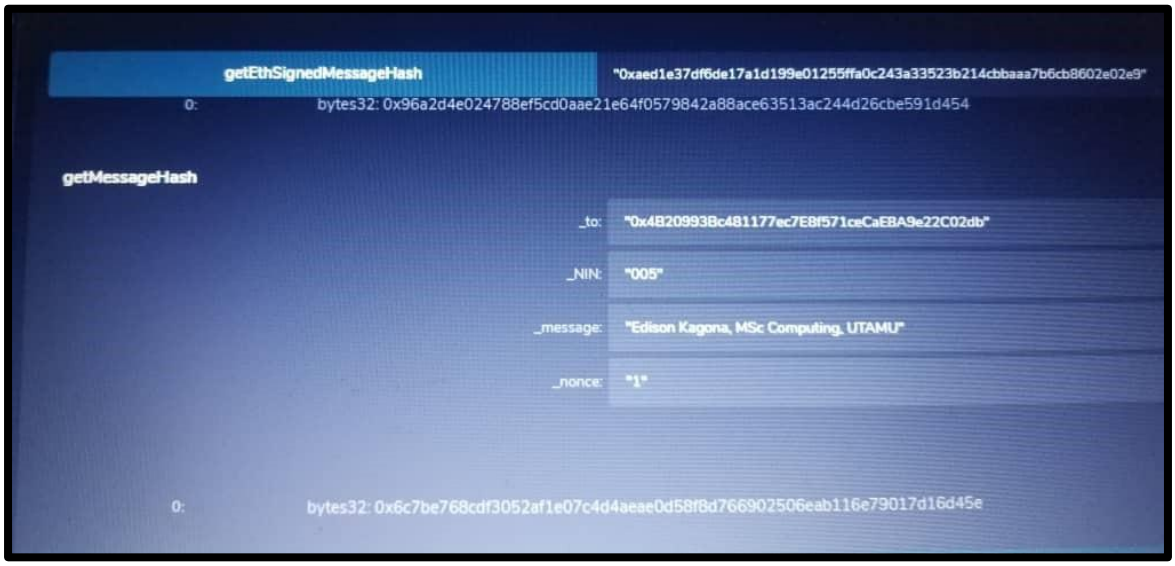


Figure 14: getMessageHash function and message structure

The message to be signed in the above figure 25 of the message structure, the getMessageHash function represents the message to be signed. We obtained a hash of the message.

bytes32: 0x6c7be768cdf3052af1e07c4d4aeae0d58f8d766902506eab116e79017d16d45e after encryption of the entire message structure.

getEthSignedMessageHash function shows the _messageHash generated in the above figure to be signed by the signer. After the signing process, we obtained a signed message

bytes32: 0x88bfa4b1ff6af3465b71e59a3ed9b1e89a68c2f10cfc179863896c3f31f8134c of the above hash.

Figure 15: Signature verification call

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

The above figure describes the signature verification call. We ran this call to verifySignature.getMessageHash function. The signature of the messages of the different accounts that represent the voters will be different.

```
Signature will be different for different accounts
0x993dab3dd91f5c6dc28e17439be475478f5635c92a56e17e82349d3fb2f166196f466c0b4e0c146f285204f0dcb13
*/
function getEthSignedMessageHash(bytes32 _messageHash) public pure returns (bytes32) {
    /*
    Signature is produced by signing a keccak256 hash with the following format:
    "\x19Ethereum Signed Message\n" + len(msg) + msg
    */
    return keccak256(abi.encodePacked("\x19Ethereum Signed Message:\n32", _messageHash));
}
```

Figure 16: Function to get the signed message hash

The result of the above function in figure is responsible for signing the message hash is shown below in figure 28 as bytes32: 0x88bfa4b1ff6af3465b71e59a3ed9b1e89a68c2f10cfc179863896c3f31f8134c. The result is the signed message hash.

Figure 17: getEthSignedMessageHash - signed message hash

The above signed message hash is written to the blockchain and later published on the entire network chain with the message owner public key. The signed message can be verified as shown in the figure below before it can be added to the blockchain network.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

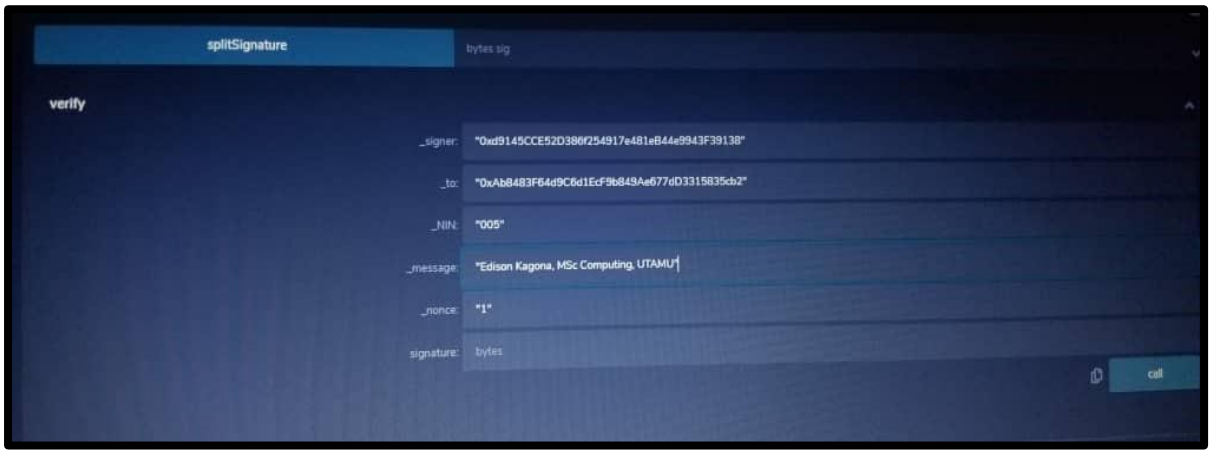


Figure 18: Signature verification

First 32 bytes stores the length of the signature,

add (sig, 32) = pointer of sig + 32

effectively, skips first 32 bytes of signature,

mload(p) loads next 32 bytes starting at the memory address p into memory

// first 32 bytes, after the length prefix

r: = mload (add (sig, 32))

// second 32 bytes

s: = mload (add (sig, 64))

// final byte (first byte of the next 32 bytes)

v: = byte (0, mload (add (sig, 96)))

// implicitly return (r, s, v)

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

4.2. Smoke Tests

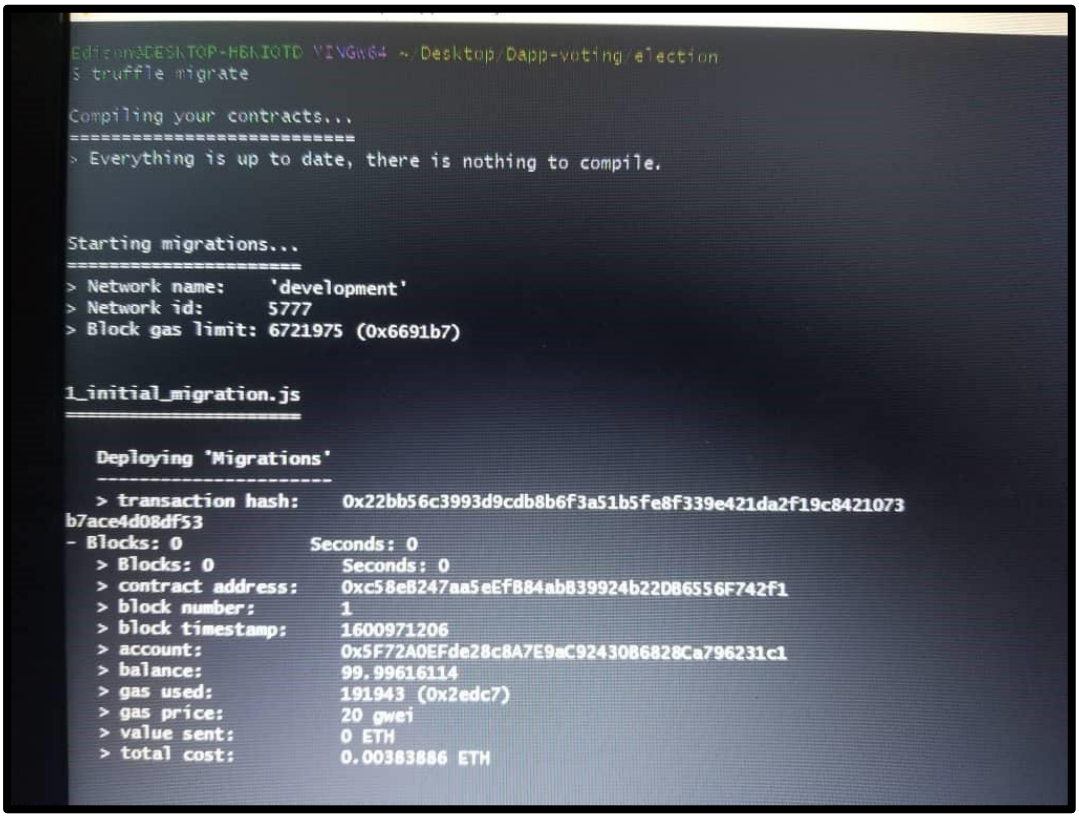


Figure 19: Running migrations

In this first smoke test after running migrations of the implemented contract called “Election”, we notice an automatic generation of a “Network name, Network ID, Block gas limit” as implemented and set up within the development environment. Deploying “Migrations,” we notice the generation of “transaction hash, blocks, contract address, block number, block timestamp” as stated in the implementation of the smart contract.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

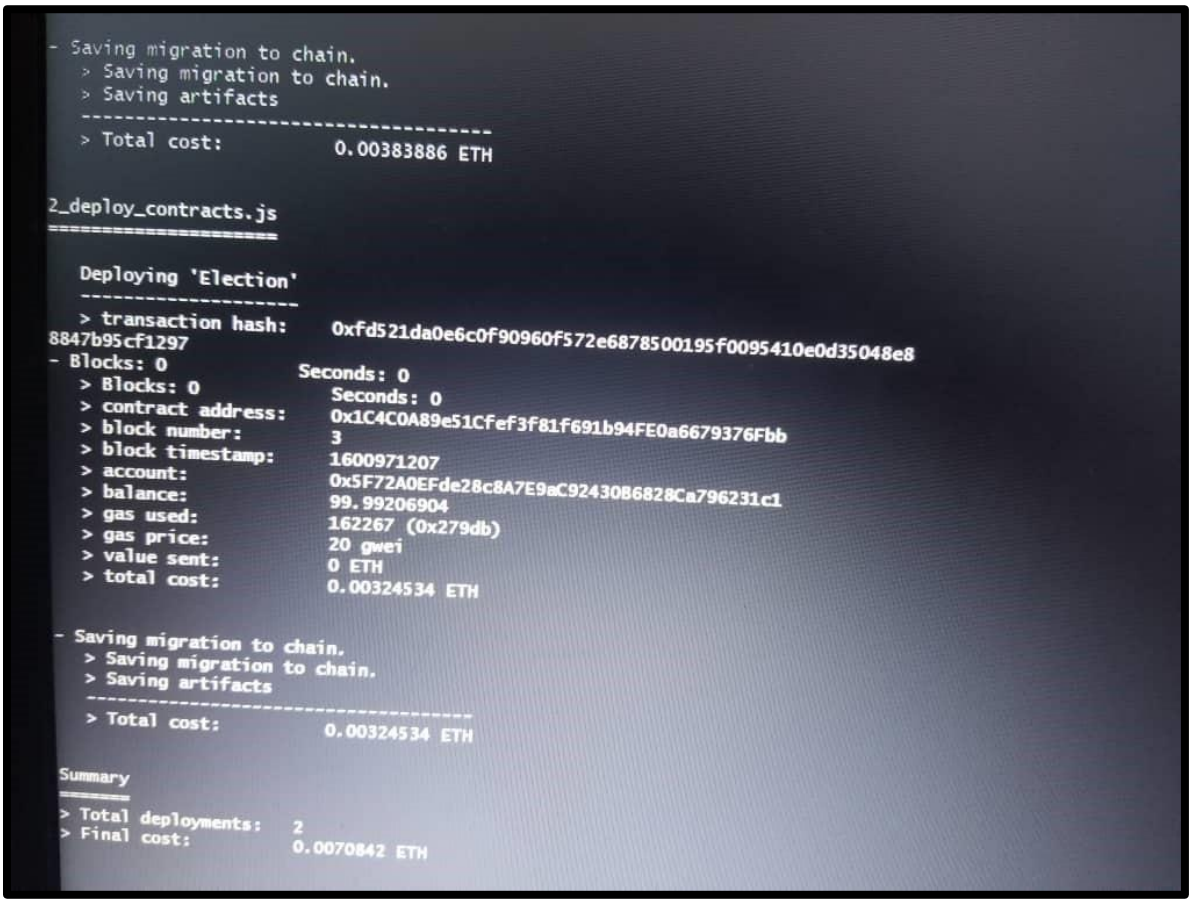


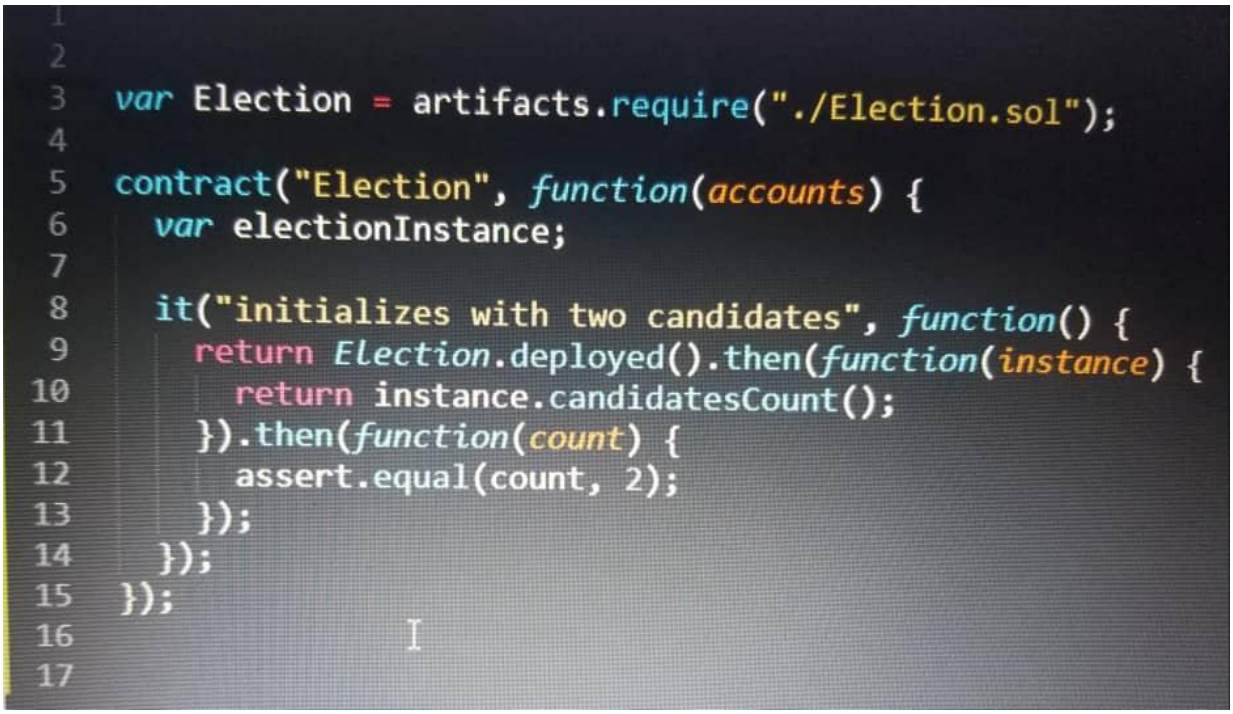
Figure 20: Deploying 'election' contract to the blockchain

The above screenshot shows the deploying of contracts (Election Contract), and the saving of all migrations to the chain, as well as a generated summary of deployments.

Tests of the Implemented Prototype

Tests were written in JavaScript with a Mocha testing framework and the Chai assertion library. These come bundled with the Truffle framework. We wrote the tests in JavaScript to simulate client-side interaction with our smart contract.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

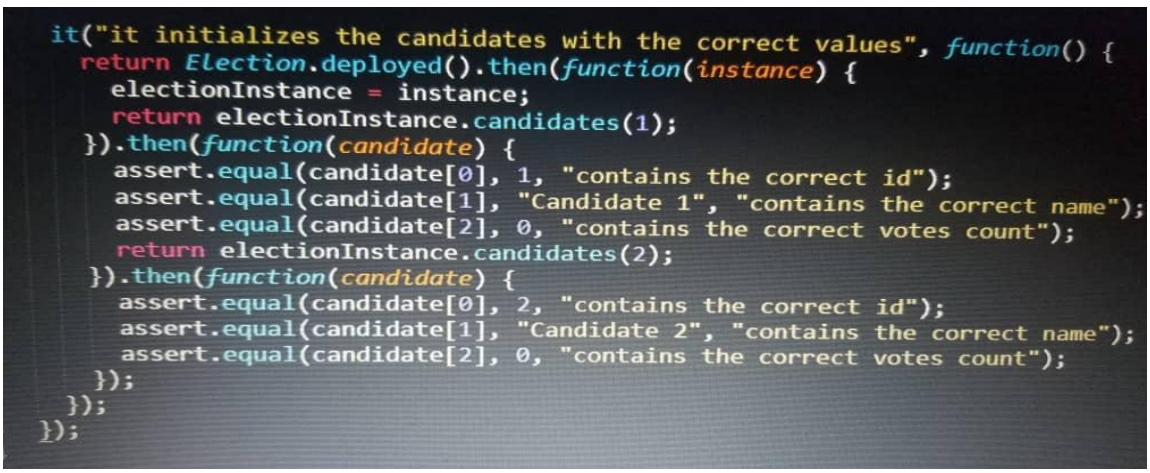


```
var Election = artifacts.require("./Election.sol");

contract("Election", function(accounts) {
  var electionInstance;

  it("initializes with two candidates", function() {
    return Election.deployed().then(function(instance) {
      return instance.candidatesCount();
    }).then(function(count) {
      assert.equal(count, 2);
    });
  });
});
```

Figure 21: Tests to simulate client-side interaction with our smart contract

In the above test, we first required the contract and assigned it to a variable. Next, we called the “contract” function and wrote all our tests within the callback function. This callback function provided an “account’s” variable that represented all the accounts on our blockchain, provided by Ganache. In the above screenshot, we check that the contract was initialized with the correct number of candidates by checking the candidate's count is equal to 2.

The next test screenshot below inspects the values of each candidate in the election, ensuring that each candidate has the correct ID, name, and vote count.



```
it("it initializes the candidates with the correct values", function() {
  return Election.deployed().then(function(instance) {
    electionInstance = instance;
    return electionInstance.candidates(1);
  }).then(function(candidate) {
    assert.equal(candidate[0], 1, "contains the correct id");
    assert.equal(candidate[1], "Candidate 1", "contains the correct name");
    assert.equal(candidate[2], 0, "contains the correct votes count");
    return electionInstance.candidates(2);
  }).then(function(candidate) {
    assert.equal(candidate[0], 2, "contains the correct id");
    assert.equal(candidate[1], "Candidate 2", "contains the correct name");
    assert.equal(candidate[2], 0, "contains the correct votes count");
  });
});
```

Figure 22: Tests to inspect the values of each candidate in the election

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

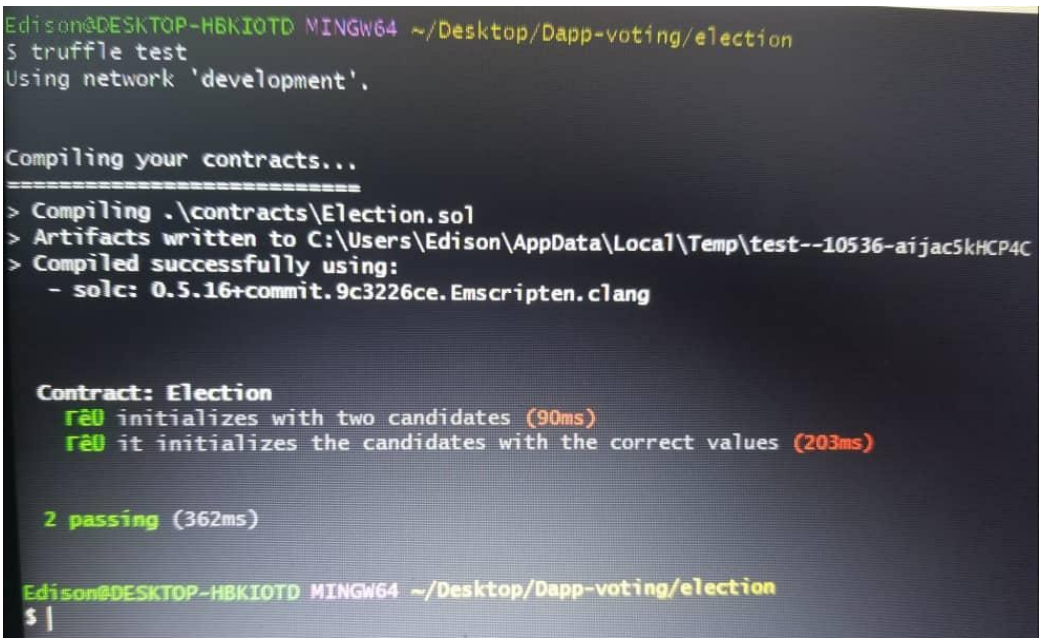


Figure 23: Illustration of the tests results

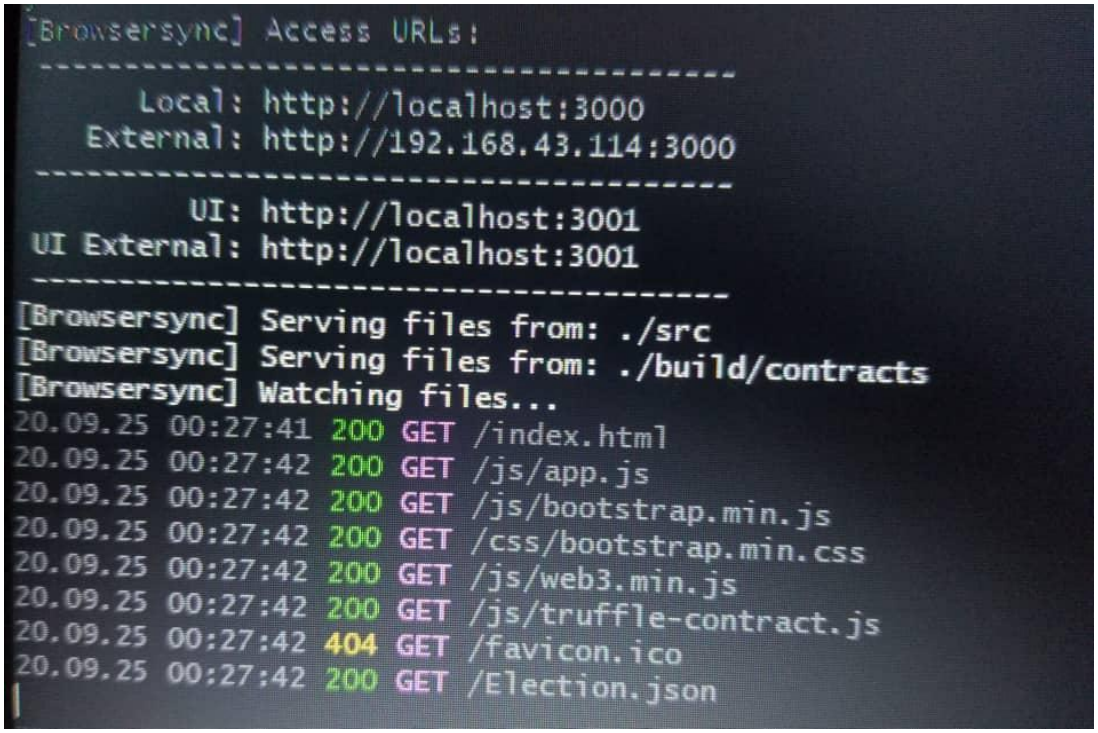


Figure 24: Running the Lite-Server

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

After starting the Lite-server on localhost:3000, a new browser window will be displayed with the client-side application. The client-side application displays “Loading …” meaning that the “Lite-server” is connected to the client-side application but not yet connected to the blockchain.

Figure 25: Lite-Server connection to the client-side application

After connecting to the blockchain, having imported one of the accounts from Ganache into Metamask, all of the contract and account data is loaded. If an account is not connected as yet, then the application will display “null”, meaning that the client-side application is running and connected to the Lite-server but no account is connected to it from the local blockchain.

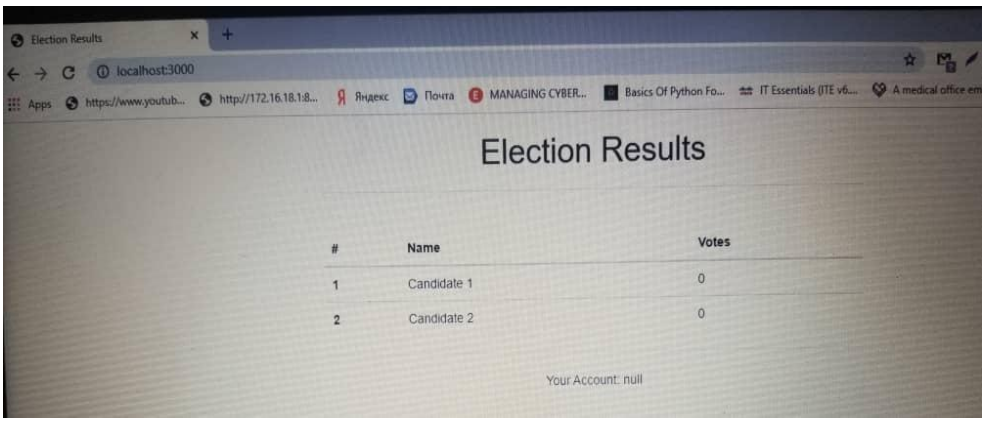


Figure 26: Null account connected

After connecting an account from Ganache, the local blockchain, the screenshot below shows a connected account to the client-side application.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

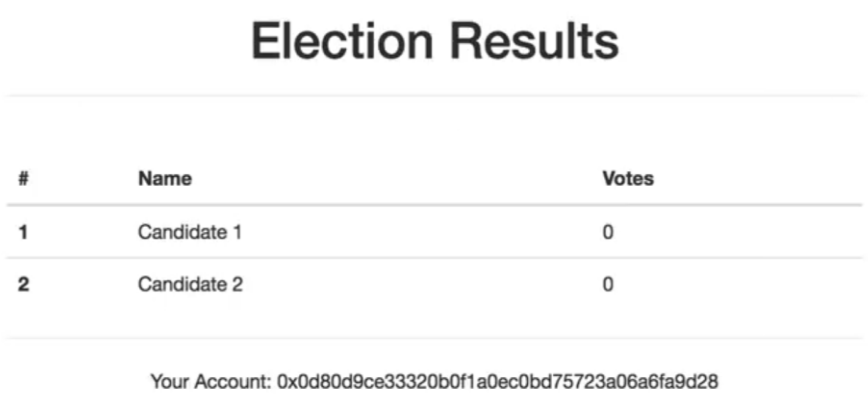


Figure 27: An account connected successfully to the client-side application

At this level, we added a way to cast votes in the election. We defined a "voters" mapping to the smart contract to keep track of the accounts that have voted in the election.

```
it("allows a voter to cast a vote", function() {
  return Election.deployed().then(function(instance) {
    electionInstance = instance;
    candidateId = 1;
    return electionInstance.vote(candidateId, { from: accounts[0] });
  }).then(function(receipt) {
    return electionInstance.voters(accounts[0]);
  }).then(function(voted) {
    assert(voted, "the voter was marked as voted");
    return electionInstance.candidates(candidateId);
  }).then(function(candidate) {
    var voteCount = candidate[2];
    assert.equal(voteCount, 1, "increments the candidate's vote count");
  })
});
});
```

Figure 28: Testing the voting function

In the above test function, we tested two things. We tested that the function increments the vote count for the candidate and that the voter is added to the mapping whenever they vote. The screenshot below shows the test function results:

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

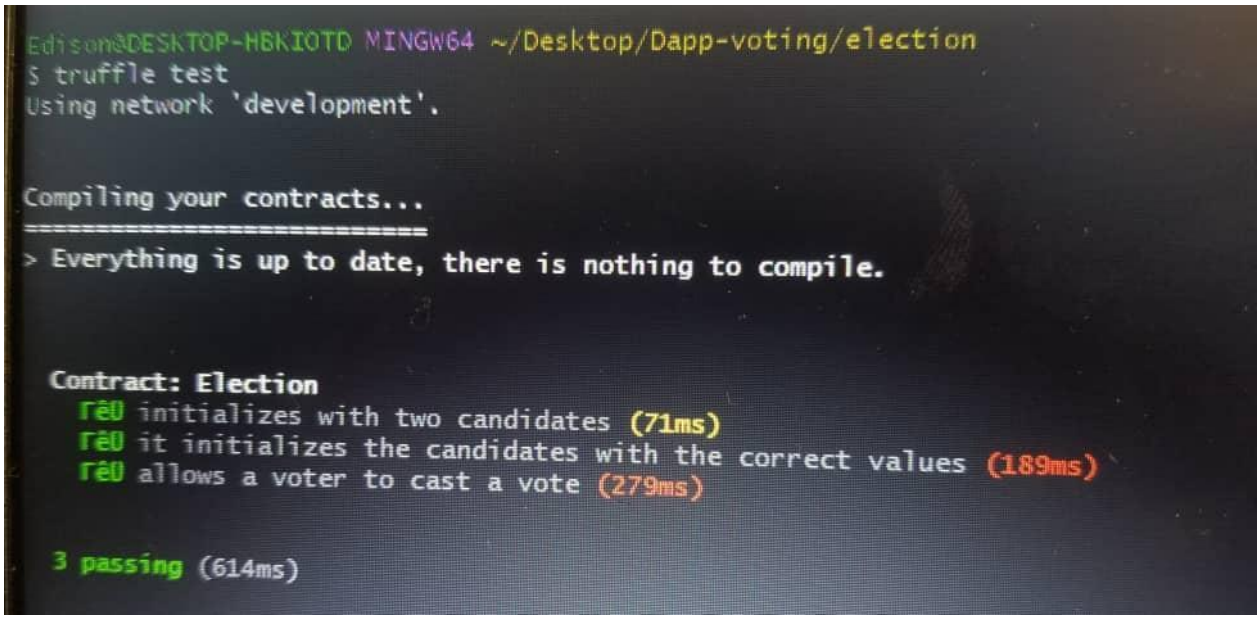


Figure 29: Voting function results

The function's requirement tests ensured that our voter functions throw an exception for double voting. We asserted that if the transaction fails, an error message is returned. We ensured that the error message contains the "revert" substring. Ensuring that our contract's state was unaltered, we made sure that the candidate did not receive any votes. The screenshot below describes the exceptions:

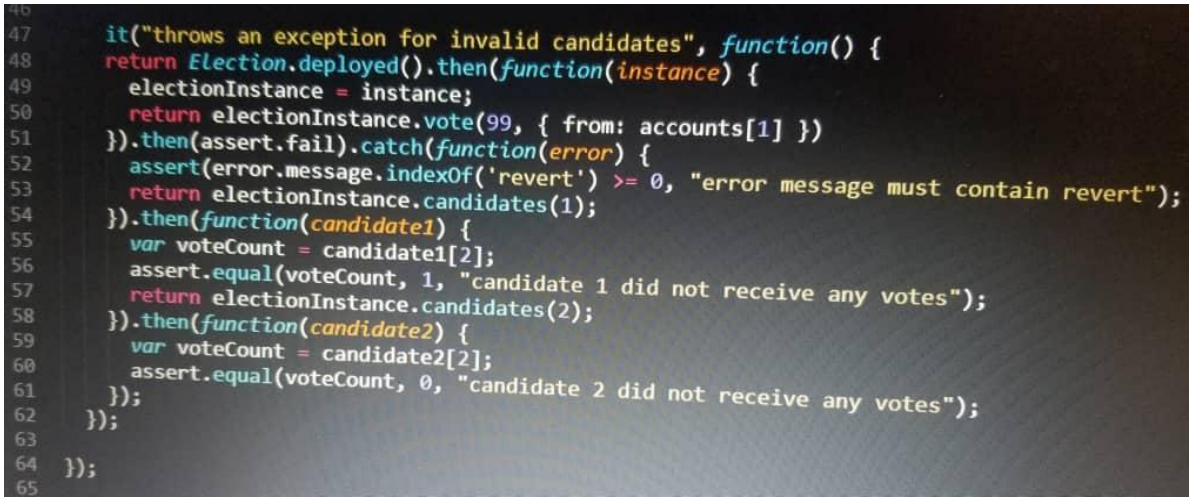


Figure 30: Exception handling for valid candidates

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

Tests to prevent double voting. The screenshot below describes the stated exceptions:

```
    it("throws an exception for double voting", function() {
    return Election.deployed().then(function(instance) {
      electionInstance = instance;
      candidateId = 2;
      electionInstance.vote(candidateId, { from: accounts[1] });
      return electionInstance.candidates(candidateId);
    }).then(function(candidate) {
      var voteCount = candidate[2];
      assert.equal(voteCount, 1, "accepts first vote");
      // Try to vote again
      return electionInstance.vote(candidateId, { from: accounts[1] });
    }).then(assert.fail).catch(function(error) {
      assert(error.message.indexOf('revert') >= 0, "error message must contain revert");
      return electionInstance.candidates(1);
    }).then(function(candidate1) {
      var voteCount = candidate1[2];
      assert.equal(voteCount, 1, "candidate 1 did not receive any votes");
      return electionInstance.candidates(2);
    }).then(function(candidate2) {
      var voteCount = candidate2[2];
      assert.equal(voteCount, 1, "candidate 2 did not receive any votes");
    });
  });

});
```

Figure 31: Exception handling for double voting

Testing the Client-Side Voting

First, we set up a test scenario with a fresh account that has not voted yet. Then we cast a vote on their behalf. Then we try to vote again using the same account to test the validating exceptions. We assert that an error occurred here, and as well inspect the error message, and ensure that no candidate received votes. The screenshot below describes the test results of the exceptions set:

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

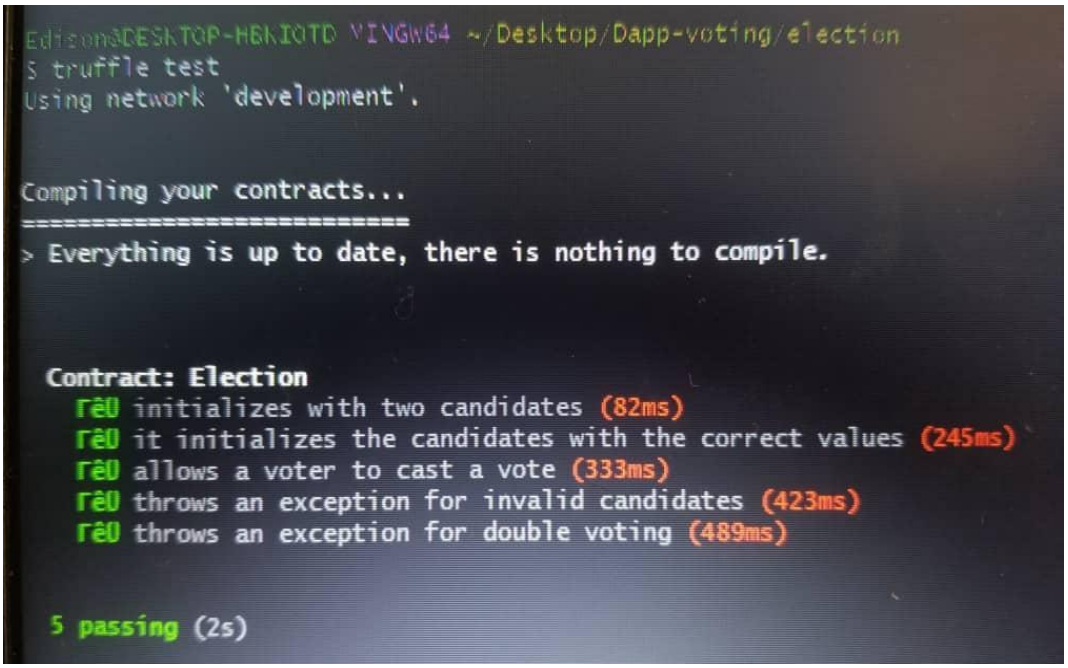


Figure 32: Testing client-side voting and exception handling

- Client-Side Front-end Interface – Screenshots Capture

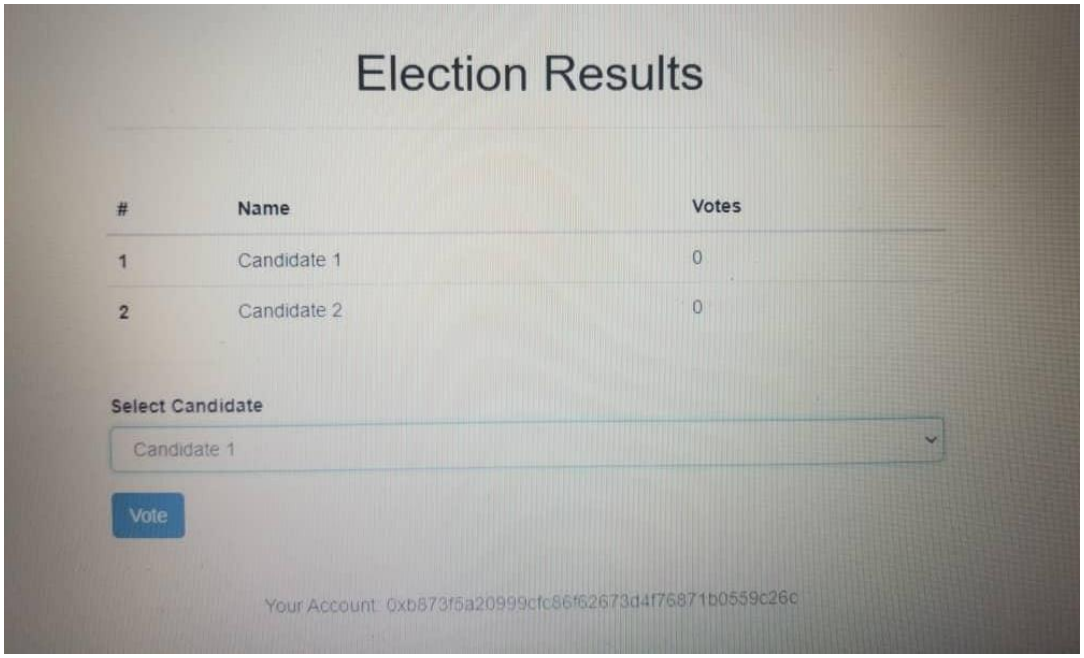


Figure 33: Single candidate display – select candidate and cast your vote

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

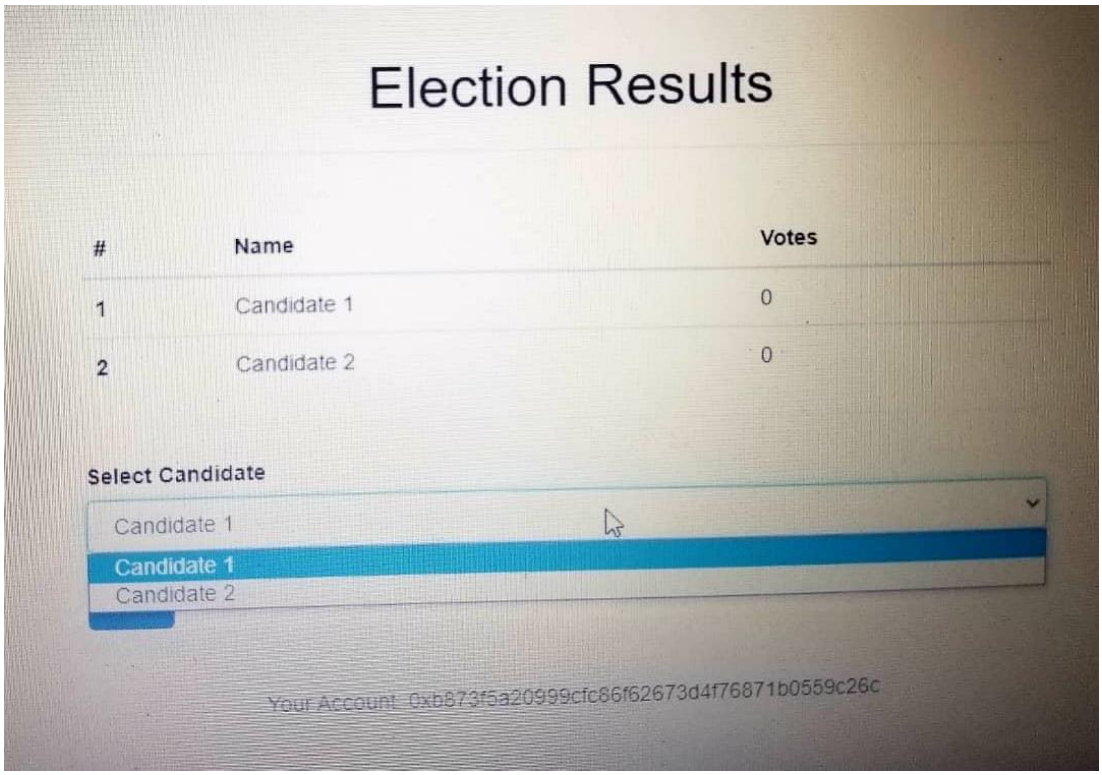


Figure 34: Display of all candidates in a drop-down

When the voting function is run, a Metamask confirmation pops up requesting you to either accept the submission, reset or reject the submission. Once one clicks submit, the vote gets cast successfully.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

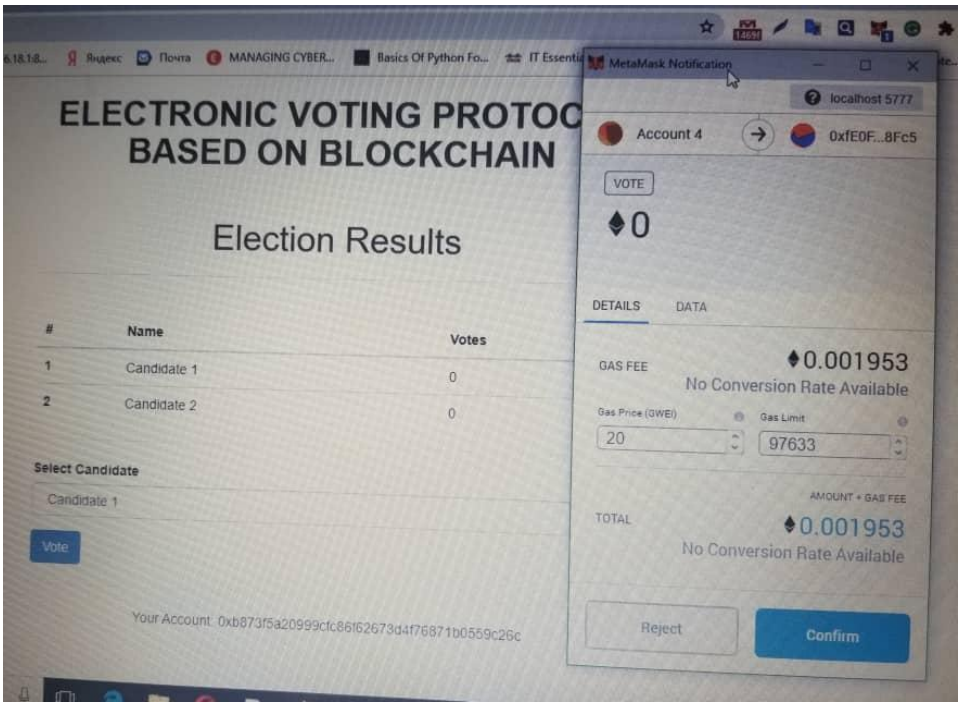


Figure 35: Metamask authentication

After the vote is taken, the "vote" button disappears for that particular voter and he or she can see and verify his or her cast vote in real time. In the screenshot below, we have tested with three voter accounts to cast the vote and observe the votes come in real time.

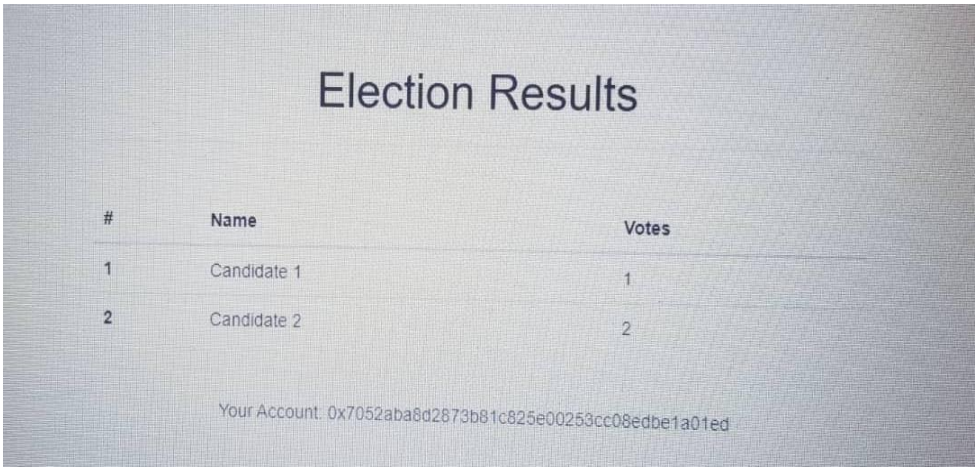


Figure 36: Real-time display of votes

In this last step, we modified to trigger an event whenever a vote is cast. This enabled us to update our client-side application when an account has voted.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

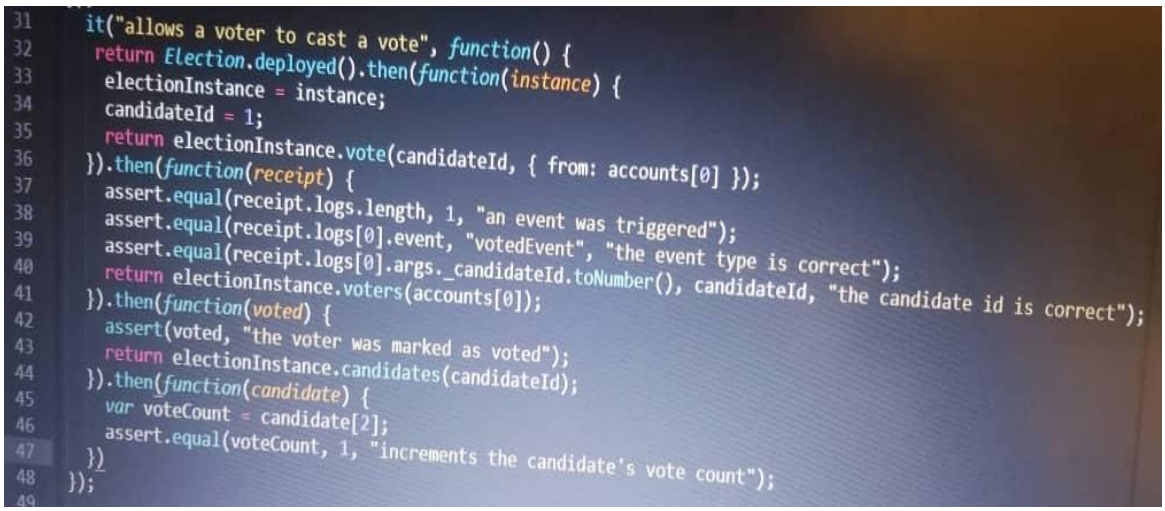


Figure 37: Checking triggered "voted" event inside our "vote" function

The above test inspects the transaction receipt returned by the "vote" function to ensure that it has logs. These logs contain the event that was triggered. We check that the event is the correct type and that it has the correct candidate ID.

To watch events in real time, we updated the client-side application to listen for the voted event and fire a page refresh anytime that it was triggered. We did that with a "listenForEvents" function. The function does a few things: First, we subscribe to the voted event by calling the "votedEvent" function. We pass in some metadata that tells us to listen to all events on the blockchain. Then we "watch" this event. Inside here, we log to the console anytime a "votedEvent" is triggered. We also re-render all the content on the page. This will get rid of the loader after the vote has been recorded and show the updated vote on the table.

SUMMARY, FUTURE WORK, DISCUSSIONS, AND CONCLUSIONS

Summary

The convolutional neural network (CNN) uses relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

The proposed system introduces a lot of desirable properties from blockchain and meets the essential requirements of an electronic voting process. All votes in the system are cryptographically linked block by block. The block with a higher value of signature is selected over others when they have the same timestamp. During elections, using the voter's choice in the election represented by CHj, a voter can make a choice that ranges from a set of predefined choices. In order to assure fairness, the voter reveals only a digital commitment ($DCHj$) over

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

the said choice. $DCHj$ (Digital commitment for CHj) only reveals the choice itself only during the counting stage of the election. The voter can vote following the list of candidates by voting for any other persons he/she prefers. Generally, the vote is public, thus the information of the vote is not encrypted. This system can be applied to a variety of voting situations and other applications. Although blockchain is a secure technology, it uses Elliptic-curve-based (ECC) public-key encryption/decryption, which is not secure to quantum computer attacks.

Future Work

This system still has a large room for improvement. When it comes to certain areas of facial recognition, training large CNNs is a resource-intensive task that requires specialized Graphical Processing Units (GPU) and highly optimized implementations to get optimal performance from the hardware. GPU memory is a major bottleneck of the CNN training procedure, limiting the size of both inputs and model architectures, and on the other side of the voting phases, improving the voter's anonymity while voting electronically on the internet. In addition, the accomplishment of multiple voting within one vote can be an ideal topic for further study, blockchain with counter-measures to quantum computer attacks, and securing electronic voting applications against client-side attacks.

Conclusion

The blockchain-based convolutional neural network e-voting system is decentralized and does not need to rely on human trust or a centralized server. The registered voter has the right to vote using their electronic devices connected to the internet. All the vote records are publicly distributed and can be verified by any intended personnel. In this system, everyone involved in the election, including spectators, who can see the voting process (recorded on the blockchain), can verify the whole election's procedure and its outcome. Each voter can verify individual voting procedures, for example, whether their ballot has been cast, recorded successfully and counted in the final tally. Dependability is guaranteed by the cryptographic algorithms and the practical consensus mechanisms of blockchain; the system protects the voting procedure against dishonest behaviors and attacks; consistency is supported by the practical consensus mechanisms of blockchain. All participants involved in the election hold the same record of the voting procedure, and thus accept the same outcome of the election. The whole voting procedure recorded on the blockchain is auditable after the election; only voters themselves know the information of their votes, and all ballots in the ballot box have no connection with their voters. Due to the transparency of blockchain, the whole procedure is open to the public and this leads to more fairness and validity.

Attackers planning to get hold of vital data assets using SQL injection have to intrude into every system to ruin the network, which is nearly impossible. Even if hackers happen to gain access to the network, the changes they make to the data through SQL injection will be reflected in the systems, notifying every participant about the same. All of these make blockchain unique and probably the best solution for any infiltrations, data tempering, and server-side SQL injection attacks.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

Discussions

In the CNN implementation, sometimes, the parameter sharing assumption may not make sense. This is especially the case when the input images to a CNN have some specific centered structure for which we expect completely different features to be learned on different spatial locations. One practical example is when the inputs are faces that have been centered in the image; we might expect different eye-specific or hair-specific features to be learned in different parts of the image. In that case, it is common to relax the parameter sharing scheme, and instead, simply call the layer a "locally connected layer".

- Privacy of Data Transmission

In our proposed system, the communication through the blockchain network may disclose voters' IP addresses, which may lead to the exposure of connections between voters and ballots via network analysis. To enhance voters' privacy, we recommend voters use anonymity services like proxies or TOR, with which voters can hide their IP addresses.

- Security Analysis

In general, the security of our system mainly relies on that of Ethereum signatures implemented using blind RSA signatures and blockchain. In the following, we discuss several security issues with this design.

a) Ballot Manipulation and Forgery

In the proposed system, manipulated ballots will be rejected by the network due to wrong signatures and incorrect formats of ballots. Meanwhile, for potential dishonest EA officers, it is impossible to return a wrong signature to invalidate voters' ballots, since wrong signatures associated with the original messages can be detected on the blockchain. When the attack aims to forge ballots, it cannot succeed if there exists at least one honest EA officer.

b) Network Attack

When there are enough honest nodes in the P2P network, the intercepted ballot will be resent, and then accepted by those honest nodes and recorded on the blockchain. Note that in our proposed design, malicious nodes can hardly influence the voting procedure. We also note that replay attacks do not work to forge multiple ballots in this design because if two ballots have the identical voting string, they will be counted only once.

c) Ballot Collision

Ballots are identified by the choice code and the random string in the voting string. If different voters produce the same string, a collision occurs and one of the two ballots will be invalid. According to the Birthday Attack, for 128-bit voting strings, the probability that collisions occur is less than 10^{-18}. Therefore, we can ignore the existence of collisions provided that the random string is long enough.

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

REFERENCES

[1] Alexander H. Trechsel, K. V. (2020). *Internet Voting in Estonia.* European UnionDemocracy Observatory.
[2] Ben Goldsmith, H. R. (2013). *Implementing and Overseeing Electronic Voting and Counting Technologies.* International Foundation for Electoral Systems and National Democratic Institute for International Affairs.
[3] Brownlee, J. (2018, 07 27). *How to Configure the Number of Layers and Nodes in a Neural Network*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/
[4] Chaum, D. (2019). *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms.* Technical Note Programming Techniques and Data Structures.
[5] Dawid Połap, G. K. (2021, 05). Agent architecture of an intelligent medical system based on federated learning and blockchain technology. *Journal of Information Security and Applications*.
[6] Fogg, k. (2002). *International Electoral Standards, Guidlines for reviewing the legal framework of Elections.* International Institute for Democracy and Electoral Assistance (International IDEA.
[7] Germann, M. (2020). *Making Votes Count with Internet Voting.* Springer.
[8] Hall, T. (2021). *Electronic voting.* Verlag Barbara Budrich.
[9] Joseph Siegle, C. C. (2021, 01 12). *Taking Stock of Africa's 2021 Elections.* Retrieved from AFRICA CENTER FOR STRATEGIC STUDIES: https://africacenter.org/spotlight/2021-elections/
[10] Kazi Sadia, M. M. (2020). *Blockchain Based Secured E-voting by Using the Assistance of Smart Contract.* . Department of IT Convergence Engineering, Kumoh National Institute of Technology, Gumi, 39177, South Korea.
[11] Kumar, D. A., & Begum, T. U. (2012). Electronic voting machine — A review. *International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012)*.
[12] Lee, D. (2020). *Vulnerabilities in Cybersecurity: U.S. Election Infrastructure.* American University washington , DC.
[13] Lewis, A. (2018). *The Basics of Bitcoins and Blockchains: An Introduction to Cryptocurren-cies and the Technology that Powers Them.*
[14] Masood Ahmad, A. U. (2020). *Security, usability, and biometric authentication scheme for electronic voting using multiple keys.* International Journal of Distributed Sensor Networks.
[15] Meredith Applegate, T. C. (2020). *Considerations on Internet Voting: An Overview for Electoral Decision-Makers.* International Foundation for Electoral Systems.
[16] Mirev. (2021, 09 28). *What is Cryptography*. Retrieved from medium.com: https://medium.com/the-capital/what-is-cryptography-part-2-blockchain-101-17ff36fcd086
[17] Mishra, M. (2020, 08 26). *Convolutional Neural Networks, Explained*. Retrieved from towardsdatascience.com: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939
[18] Peter Loshin, M. C. (2021). *Data security guide: Everything you need to know*. Retrieved from searchsecurity.techtarget.com: https://searchsecurity.techtarget.com/definition/encryption

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ

[19] Ricardo Ñanculef, S. B. (2020). *Training Convolutional Nets to Detect Calcified Plaque in IVUS Sequences*. Retrieved from www.sciencedirect.com: https://www.sciencedirect.com/topics/engineering/convolutional-layer
[20] Robert S.Mueller, E. B. (2020). *The impeachment of Donald Trump.* e-artnow.
[21] Schwartz, J. (2018, 11 01). *The Vulnerabilities of Our Voting Machines*. Retrieved from www.scientificamerican.com: https://www.scientificamerican.com/article/the-vulnerabilities-of-our-voting-machines/
[22] Shinder, L. (2008). *Understanding Cybercrime Prevention*. Retrieved from www.sciencedirect.com: https://www.sciencedirect.com/topics/computer-science/secret-key-encryption
[23] Somayeh Dolatnezhad, M. A. (2019). *Preventing SQL Injection Attacks by Automatic Parameterizing Raw Queries Using Lexical and Semantic Analysis Methods.* Scientia Iranica.
[24] Sonam Panda, R. S. (2013). *Protection of Web Application against Sql Injection Attacks.* International Journal of Modern Engineering Research (IJMER).
[25] Vadakkanmarveettil, J. (2021, 02 18). *What is Blind SQL Injection? A Simple Overview*. Retrieved from www.jigsawacademy.com: https://www.jigsawacademy.com/blogs/cyber-security/blind-sql-injection
[26] Wang, Y. (2017). *An E-voting Protocol Based on Blockchain.* .
[27] Wei, C. C. (2020). *Blockchain-Based Electronic Voting Protocol.* Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia, Malaysia.
[28] Wenlei Qu, L. W. (2020). *A electronic voting protocol based on blockchain and homomorphic signcryption.* ETRI Journal.
[29] Wolf, P. (2017). *Introducing Biometric Techmology in Elections.* International Institute for Democracy and Electoral Assistance.
[30] Zhonghua Zhang, X. S. (2021, 03 19). *Recent Advances in Blockchain and Artificial Intelligence Integration: Feasibility Analysis, Research Issues, Applications, Challenges, and Future Work.* Retrieved from www.hindawi.com: https://www.hindawi.com/journals/scn/2021/9991535/
[31] Zur Erlangung des Grades, D.-I. (.-I. (2010). *Privacy and Verifiability in Electronic Voting.*

Article DOI: 10.52589/AJSTE-X4ASIQPQ
DOI URL: https://doi.org/10.52589/AJSTE-X4ASIQPQ