

Voting on a Blockchain

Marisa Paryasto

25 Nov 2019

Basic Principles of Voting

- **Secret Ballot:** Your vote is secret. Nobody should be able to link your vote back to your race, gender, age and personal profile.
- **One man, one vote:** Every voter votes once and the voting system must be able to reconcile the total number of votes to the total number of voters and those who did not vote.
- **Voter eligibility:** Only eligible voters are allowed to vote.
- **Transparency:** The vote counting process is fixed, rules are well established, known to voters and withstands public scrutiny.
- **Votes accurately recorded and counted:** Vote counting is consistent. Rules are cast in stone. Vote counts are audit-able.
- **Reliability:** Voting system must be accurate and verifiable. Safeguards are in place to prevent frauds, accidents and security breaches.

1 - Create MetaMask Accounts

create 4 accounts in MetaMask to represent the following participants:

- The Chairman — manages the ballot process
- Zoe — voter
- Ellie — voter
- Peter — voter

Copy and paste Ballot.sol (<https://github.com/jacksonng77/ballot>) to remix.ethereum.org.

The image shows the Remix IDE interface. The left pane displays the Solidity code for the `Ballot` contract. The right-hand sidebar contains deployment and interaction options.

Contract Definition: `Ballot` (0 reference(s))

```
1  /**
2   * @file ballot.sol
3   * @author Jackson Ng <jacksonn@tp.edu.sg>
4   * @date 21st Apr 2019
5   */
6
7  pragma solidity ^0.5.0;
8
9  contract Ballot {
10
11     struct vote{
12         address voterAddress;
13         bool choice;
14     }
15
16     struct voter{
17         address voterAddress;
18         string voterName;
19         bool voted;
20     }
21
22     uint public countResult = 0;
23
24     address public ballotOfficialAddress;
25     string public ballotOfficialName;
26     string public proposal;
27
28     vote[] private votes;
29     voter[] public voterRegister;
30
31     enum State { Created, Voting, Ended }
32     State public state;
33
34     //creates a new ballot contract
35     constructor(
36         string memory _ballotOfficialName,
37         string memory _proposal) public {
38         ballotOfficialAddress = msg.sender;
39         ballotOfficialName = _ballotOfficialName;
40         proposal = _proposal;
41     }
42 }
```

Environment: Injected Web3 (Custom (971693207))

Account: [Account Address]

Gas limit: 3000000

Value: 0 wei

Contract: Ballot

Deploy: string _ballotOfficialName, string _proposal

or

At Address: Load contract from Address

Transactions recorded: 0

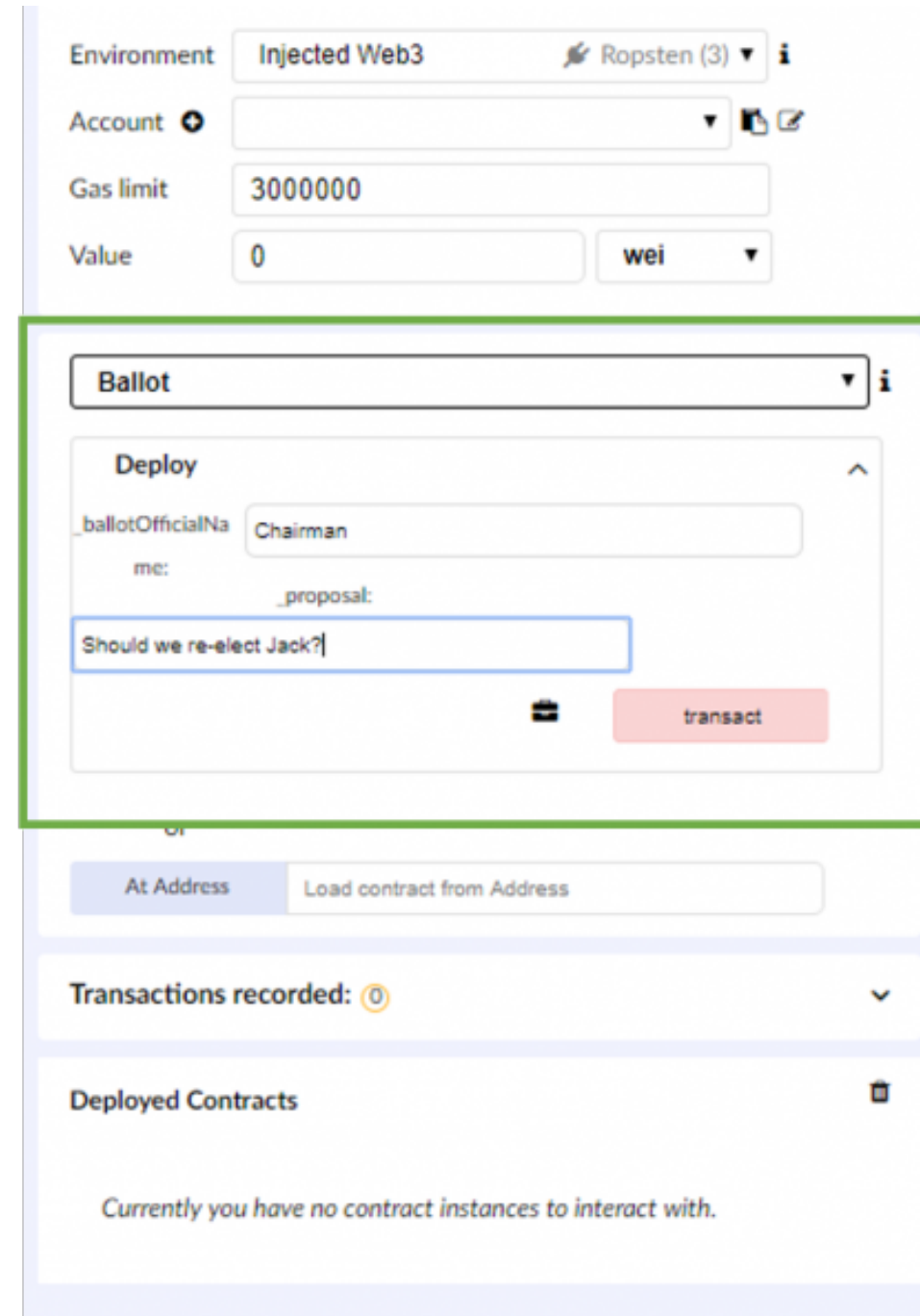
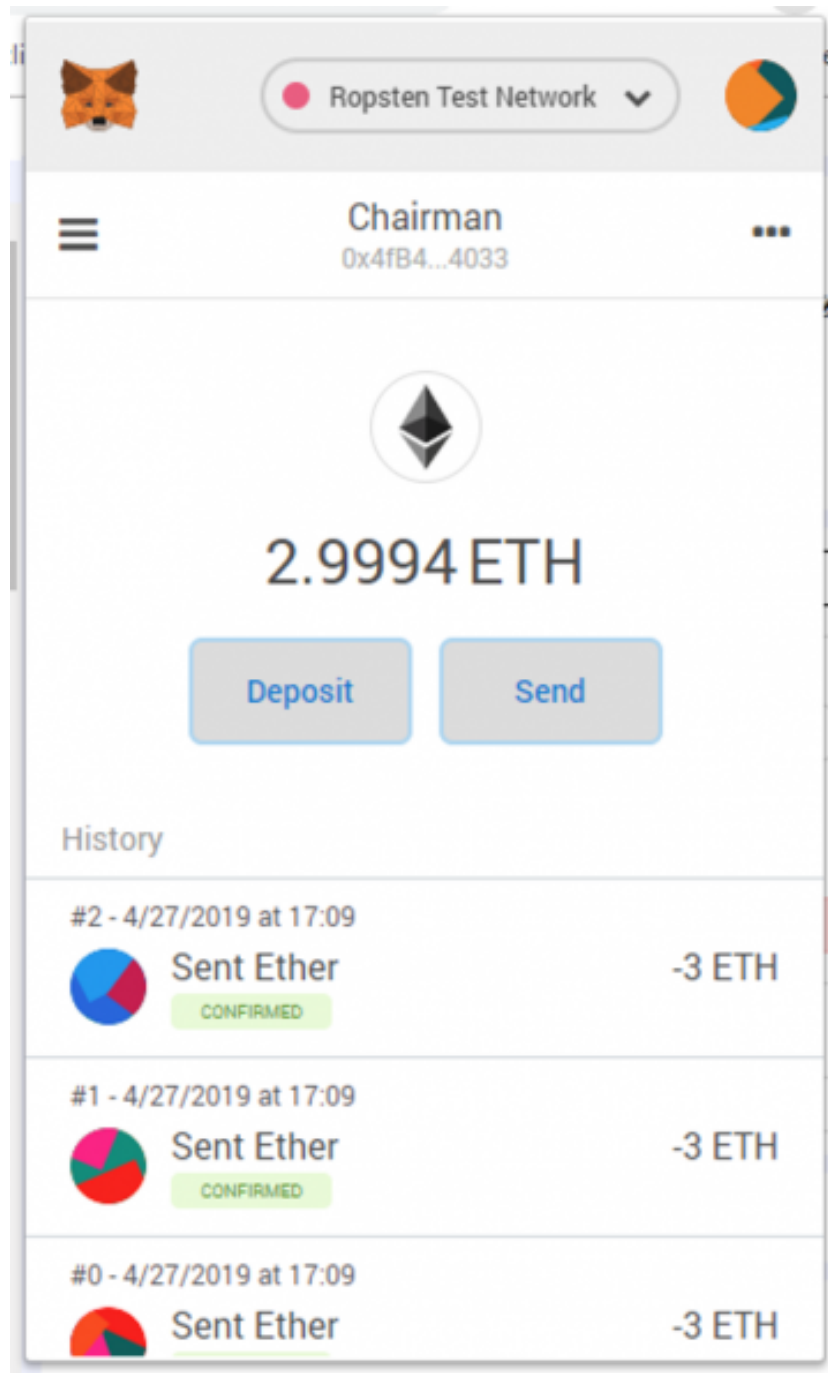
Deployed Contracts:

Currently you have no contract instances to interact with.

[2] only remix transactions, script

Search transactions

In Remix, switch to the chairman's MetaMask wallet account to deploy the Ballot contract.





The chairman adds the wallet addresses of eligible votes the voters array.



Ballot Contract

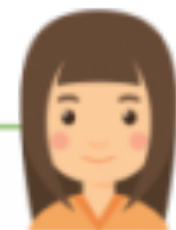


Proposal: Should we re-elect Jack?

Voters Array



0x123.... (Zoe's wallet)



0x14a.... (Ellie's wallet)

0x29b.... (Peter's wallet)



In Remix, switch to the Chairman's wallet address. Then add the wallet addresses and names of Zoe, Ellie and Peter using the addVoter function.

The image shows a screenshot of the Remix IDE interface. On the left, the 'Chairman' wallet is displayed with a balance of 2.9994 ETH. The wallet address is 0x4fB4...4033. Below the balance, there are 'Deposit' and 'Send' buttons. A 'History' section shows three transactions, all labeled 'Sent Ether' and '-3 ETH', with a 'CONFIRMED' status.

On the right, three transaction windows are shown, each titled 'Ballot at 0x3a5...6fd4e (blockchain)'. Each window contains an 'addVoter' function call with the following details:

- Transaction 1:** _voterAddress: 0xAF703Ae5e51159A97623118dd2b0202561856787, _voterName: Zoe
- Transaction 2:** _voterAddress: 0x8A53009Aad8AED45DE3f1dF895b1f216A10c1D7F, _voterName: Ellie
- Transaction 3:** _voterAddress: 0xd03732F113392Fb5a757c99B007B7758e4F75dC2, _voterName: Peter

Each transaction window has a 'transact' button.

At any time you can click on the respective buttons below to read the ballot contract's public variables which include information about who the Chairman is, what the proposal says, the current state of the contract and the names of people in the voters register and whether or not they have voted. This ensures transparency. The chairman triggers voting to start. At this point, no new voters can be added and the proposal is cast in stone. The chairman sends the ballot contract address to eligible votes.

countVote

doVote

startVote

ballotOfficialAddresses

0: address: 0x4fB430df81d77E96483238e2BCc8884677624033

ballotOfficialName

0: string: Chairman

countResult

0: uint256: 0

proposal

0: string: Should we re-elect Jack?

state

0: uint8: 0

voterRegister

0: address: voterAddress
0xAF703Ae5e51159A97623118dd2b0202561856787
1: string: voterName Zoe
2: bool: voted false

Anyone can read the chairman's wallet address and his name.

Anyone can also read the proposal

Anyone can find out the current state of this ballot. It says 0, which means that the voting hasn't started

To find out who's in the voterRegister, enter it's position (0 for the first person in the register) and it tells you who the person is and whether he has voted

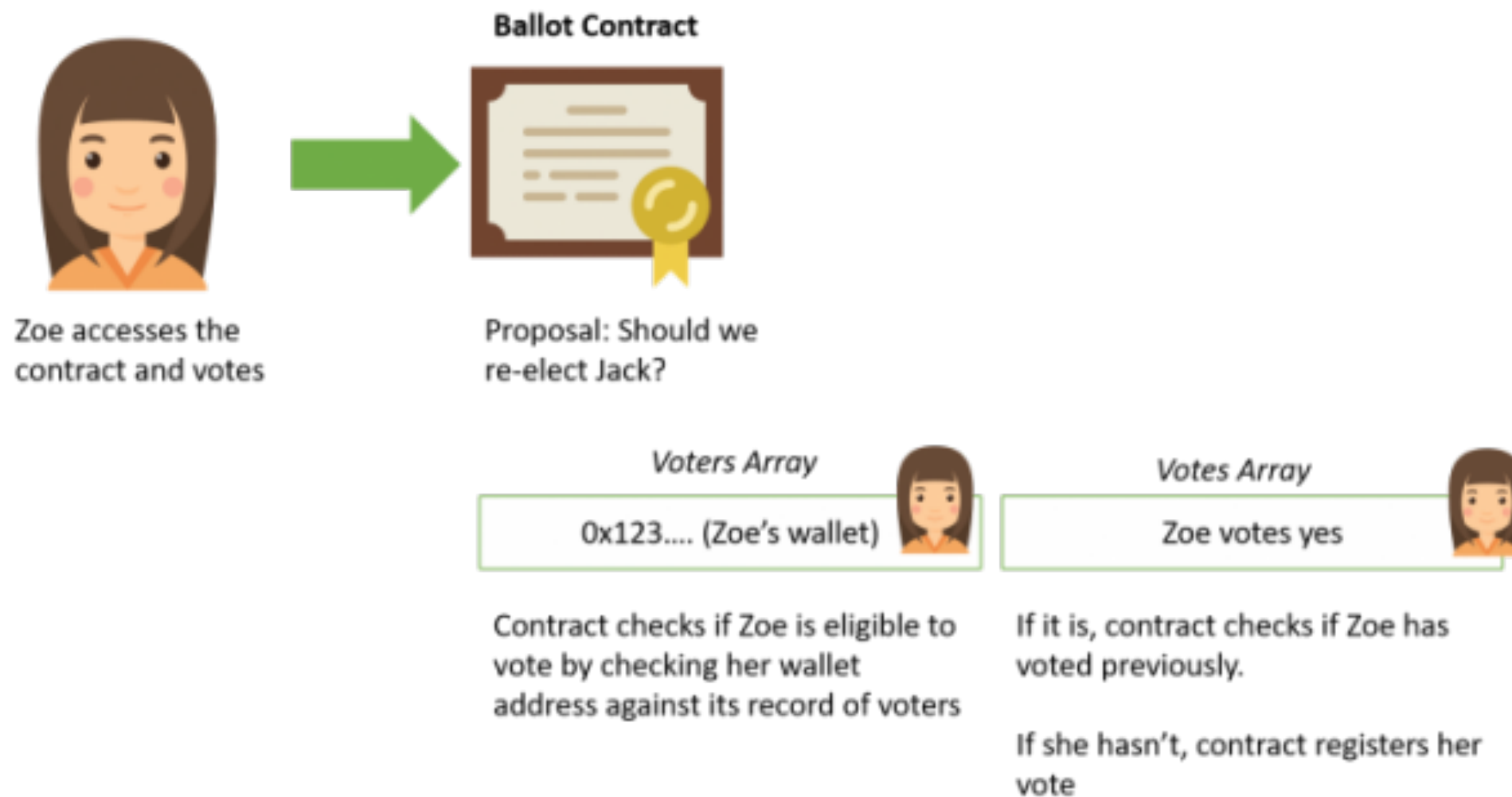
In Remix, make sure that you are still using the Chairman's wallet address. Press [startVote] to let voting begin (the contract will not allow anyone else other than the person who create it to startVote).

The image shows a screenshot of the Remix IDE interface. On the left, a wallet for 'Chairman' (address 0x4fB4...4033) is displayed on the Ropsten Test Network. The wallet balance is 2.9994 ETH. Below the balance are 'Deposit' and 'Send' buttons. A 'History' section shows three transactions, all labeled 'Sent Ether' for -3 ETH, each with a 'CONFIRMED' status and a date of 4/27/2019 at 17:09.

On the right, a smart contract is shown with several functions and variables:

- countVote**: A function button.
- doVote**: A function button with a dropdown menu set to `bool _choice`.
- startVote**: A function button, highlighted with a green border.
- ballotOfficialAddresses**: A variable with a value of `0: address: 0x4fB430df81d77E96483238e2BCc8884677624033`.
- ballotOfficialName**: A variable with a value of `0: string: Chairman`.
- countResult**: A function button with a value of `0: uint256: 0`.
- proposal**: A variable with a value of `0: string: Should we re-elect Jack?`.
- state**: A variable with a value of `0: uint8: 0`.
- voterRegister**: A function button with a dropdown menu set to `uint256`.

Zoe attempts to vote. The contract first compares Zoe's MetaMask wallet address to her record in the voters' array. If Zoe's contract address is not found in the array, she's deemed an ineligible voter and her vote will not be accepted. If Zoe's wallet address is found in the voters' array, the contract checks the vote array to confirm if she has voted previously. If she has, the ballot contract will refuse to accept her vote. If she hasn't, her vote is accepted, and her status is updated to say that she has voted. Once she has voted, the contract will no longer allow Zoe to vote again.



To vote yes, type “true” at “doVote” and click [transact].

Note that the state of the contract now says “1”, which means that voting is now ongoing.

The image shows a screenshot of a web application interface. On the left is a wallet view for 'Zoe' (0xAF70...6787) on the 'Ropsten Test Network'. It displays a balance of '3 ETH' and buttons for 'Deposit' and 'Send'. Below this is a 'History' section stating 'You have no transactions'. On the right is a smart contract interface titled 'Ballot at 0x3a5...6fd4e (blockchain)'. It features several functions: 'addVoter' (with parameters 'address _voterAddress, string _voterName'), 'countVote', 'doVote' (with a '_choice:' input field containing 'true' and a 'transact' button), 'startVote', 'ballotOfficialAddresses' (with value '0: address: 0x4fB430c...'), 'ballotOfficialName' (with value '0: string: Chairman'), 'countResult' (with value '0: uint256: 0'), 'proposal' (with value '0: string: Should we re...'), and 'state' (with value '0: uint8: 1'). Two green callout boxes provide context: one points to the 'doVote' section with the text 'Zoe votes “true” which means “yes”', and another points to the 'state' field with the text 'State says 1, which means that voting has begun'.

Wallet Interface (Left):

- Network: Ropsten Test Network
- Address: Zoe (0xAF70...6787)
- Balance: 3 ETH
- Buttons: Deposit, Send
- History: You have no transactions

Smart Contract Interface (Right):

Ballot at 0x3a5...6fd4e (blockchain)

- addVoter: address _voterAddress, string _voterName
- countVote
- doVote: [transact]
- startVote
- ballotOfficialAddresses: 0: address: 0x4fB430c...
- ballotOfficialName: 0: string: Chairman
- countResult: 0: uint256: 0
- proposal: 0: string: Should we re...
- state: 0: uint8: 1

Callouts:

- Zoe votes “true” which means “yes”
- State says 1, which means that voting has begun

Now Ellie and Peter vote.




Ballot Contract



Ellie and Peter votes too.

Proposal: Should we
re-elect Jack?

Voters Array

	0x123.... (Zoe's wallet)	
	0x14a.... (Ellie's wallet)	
	0x29b.... (Peter's wallet)	


Votes Array

	Zoe votes yes	
	Ellie votes no	
	Peter votes yes	


Change the wallet address to Ellie's and Peter's for them to cast their votes.


The image shows two side-by-side screenshots of a web application. The left screenshot displays a wallet interface for a user named 'Ellie' on the 'Ropsten Test Network'. The wallet address is '0x8A53...1D7F' and the balance is '3 ETH'. There are 'Deposit' and 'Send' buttons. Below this, a 'Queue (1)' section shows a 'Contract Interaction' for '#0 - 4/27/2019 at 17:38' with a value of '-0 ETH' and a 'PENDING' status. The bottom section is labeled 'History' and says 'You have no transactions'.

The right screenshot shows a 'Ballot at 0x3a5...6fd4e (blockchain)' interface. It has a dropdown menu with 'addVoter' and 'countVote' options. The 'doVote' section is highlighted with a green box and contains a '_choice:' field with the value 'false'. Below this is a 'transact' button. A green callout box points to the 'doVote' section with the text: 'Ellie votes "false" which means "no"'. Below the 'doVote' section, there are several fields: 'startVote', 'ballotOfficialAddresses' (0: address: 0x4fB430df81d77E96483238e2BCc8884677624033), 'ballotOfficialName' (0: string: Chairman), 'countResult' (0: uint256: 0), 'proposal' (0: string: Should we re-elect Jack?), and 'state' (0: uint8: 1).




Ropsten Test Network






Peter

0xd037...5dC2






3 ETH

Deposit

Send

Queue (1)

#0 - 4/27/2019 at 17:40



Contract Interaction

PENDING

-0 ETH

History

You have no transactions

Ballot at 0x3a5...6fd4e (blockchain)

addVoter

address_voterAddress, string_voterName

countVote

doVote

_choice: true

transact

startVote

ballotOfficialAddress

0: address: 0x4fB430df81d77E96483238e2BCc8884677624033

ballotOfficialName

0: string: Chairman

countResult

0: uint256: 0

proposal


0: string: Should we re-elect Jack?

state


0: uint8: 1


Peter votes "true" which means "yes"

At any point, anyone can check voteRegister to see if there's someone in the register who hasn't voted.





Ropsten Test Network





Chairman
0x4fB4...4033






2.9994 ETH

DepositSend

History


#2 - 4/27/2019 at 17:09

 Sent Ether

CONFIRMED

-3 ETH


#1 - 4/27/2019 at 17:09

 Sent Ether

CONFIRMED

-3 ETH

#0 - 4/27/2019 at 17:09

 Sent Ether

-3 ETH

startVote

ballotOfficialAddresses
0: address: 0x4fB430df81d77E96483238e2BCc8884677624033

ballotOfficialName
0: string: Chairman

countResult
0: uint256: 0

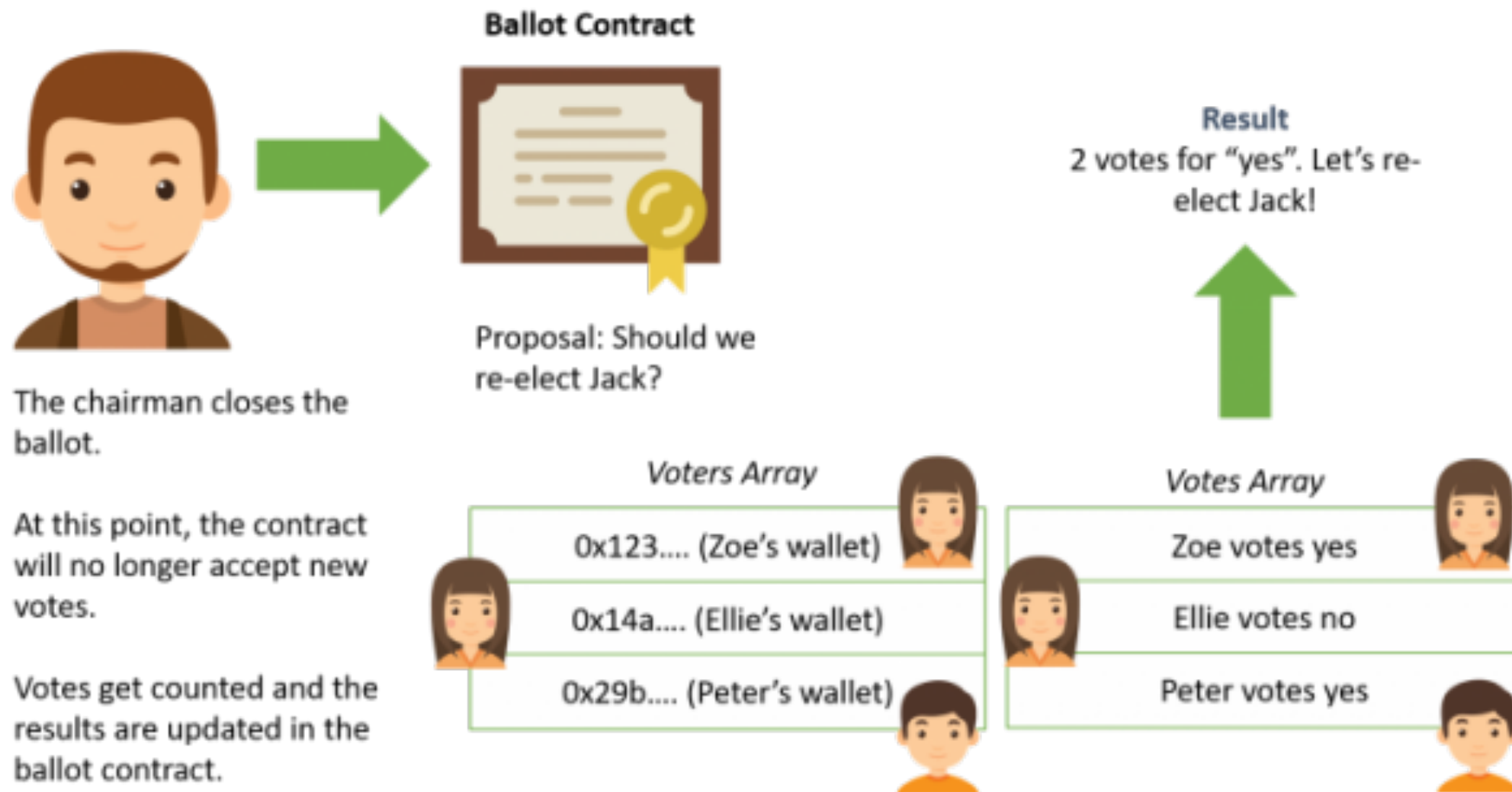
proposal
0: string: Should we re-elect Jack?

state
0: uint8: 1

voterRegister 2
0: address: voterAddress
0xd03732F113392Fb5a757c99B007B7758e4F75dC2
1: string: voterName Peter
2: bool: voted true

Check the voter register. Note that those who voted has an indication on their voter register record saying "voter – true"

Time's up. The chairman closes the ballot. Closing the ballot is a significant step because it stops allowing anyone who hasn't voted to vote. Counting of votes is not possible unless the ballot closes. The Ballot contract now counts all the votes it has received and records the total number of "yes" votes. Once this is done, everyone can view the result. The balloting process has ended and no one, not even the chairman is allowed to change any properties of the ballot contract.



To do so in Remix, switch back to the Chairman's wallet account and press [countVote]. The contract will not allow anyone other than the contract creator (which is the chairman) to execute this step. Once counting is done, results are released and the contract's state becomes 2 which means that voting has ended.

The screenshot displays the Remix IDE interface. On the left, the 'Chairman' wallet is shown with a balance of 2.9994 ETH. The right panel shows the contract's state and functions. The 'countVote' function is highlighted, and a callout box instructs the user to 'Press [countVote] to end the voting and start counting.' Below this, the 'startVote' function is shown with the 'ballotOfficialAddress' set to '0: address: 0x4fB430df81d77E96483238e2BCc8884677624033' and the 'ballotOfficialName' set to '0: string: Chairman'. The 'countResult' function is also highlighted, showing the result '0: uint256: 2'. A callout box states 'Results are released. 2 person voted "YES"'. Finally, the 'state' variable is highlighted, showing '0: uint8: 2', with a callout box stating 'State is now 2, which means that voting has ended and result is cast in stone'.

Chairman
0x4fB4...4033

2.9994 ETH

Deposit Send

History

- #2 - 4/27/2019 at 17:09
Sent Ether -3 ETH
CONFIRMED
- #1 - 4/27/2019 at 17:09
Sent Ether -3 ETH
CONFIRMED
- #0 - 4/27/2019 at 17:09
Sent Ether -3 ETH

addVoter address_voterAddress, string_voterName

countVote

doVote
_choice: true

transact

startVote

ballotOfficialAddress
0: address: 0x4fB430df81d77E96483238e2BCc8884677624033

ballotOfficialName
0: string: Chairman

countResult
0: uint256: 2

proposal
0: string: Should we re-elect Jack?

state
0: uint8: 2

Press [countVote] to end the voting and start counting.

Results are released. 2 person voted "YES"

State is now 2, which means that voting has ended and result is cast in stone

The principles of voting implemented

One man one vote

The voters array stores a list of voters who have voted. It ensures that no one can vote the second time. Once a voter votes, his status changes to “voted” and the ballot contract checks to ensure that he does not vote again.

Voter eligibility

Voter eligibility is determined by assembling a voters’ array of wallet address before voting begins. You need to vote with your MetaMask Wallet whose address matches the one that the chairman registers before voting begins.

Wait a moment, doesn’t that mean that someone who have stolen my MetaMask wallet will be able to vote on my behalf?

That’s correct, on a Blockchain, the only thing that separates you from an identity thief is your wallet’s private key.

But how does the chairman ensure that it’s really me who’s casting the vote? Facial recognition? That’s definitely a possibility — build a ballot app to allow the wallet to be unlocked to vote only if you pass a fingerprint or facial recognition test will work.

Transparency

This is one of the things that Blockchain does extremely well. Every action taken and every record etched on the Blockchain is immutable. On a public Blockchain, collusion is close to impossible as described here.

Votes accurately recorded and counted

In the ballot smart contract, the ballot goes through several states, from the point it is created, open for voting to the point where ballot is closed and votes are counted.

In each of these states, the contract dictates what the chairman and voters is allowed or not allowed to do. For example, the contract does not allow voting to start until the chairman kick-starts the voting process. It does not allow the chairman to stuff the voting box with new voters once voting begins.

Reliability

There is no single point of failure on a Blockchain as every node in the chain participates in keeping the Blockchain running. A smart contract, once deployed on the Blockchain is immutable. The business logic of the smart contract once deployed is cast in stone. There’s no way the chairman can change the rules, say, from a one man one vote to one man two vote once the contract is deployed.