Verum Electus

Blockchain Election Voting

Spring 2018 | Blockchain

Lucas Nicodemus lucas@hakusa.ro
Raylyn Pettigrew rpettig1@uwyo.edu

Abstract

Due to recent controversies regarding election integrity, we believe that the State of Wyoming needs a secure and efficient program for modernizing the election system in the state. Therefore, we propose an Ethereum smart contract that allows Wyomingite voters to cast traceable, verifiable, and secure votes on a modern, proven platform. Using a Ruby on Rails application we provide an interoperability layer to interface with the Ethereum network for developers and to ensure voter accountability. Our smart contract provides a secure platform for casting and tallying a multitude of ballots and includes write-in vote capabilities.

Background

Recent controversies regarding the United States national election have led many U.S. citizens to gain insecurities about the safety of the entire election process. However, while the trust regarding election methodologies fell, the trust in blockchain technologies rose; Wyoming became an oasis for blockchain companies when it passed H.B. 0070 and H.B. 0019 to amend current transaction laws. These two dynamic shifts in culture created a unique opportunity for a modern adaptation of an outdated system.

Courtesy of: Rampell, Catherine at *The Washington Post*. 2018

Project Highlights

Throughout our entire process, our two-person team was able to implement a voting tool using agile development techniques where we:

- Created a system for writing in a candidates name that does not appear on the official ballot.
- Implemented a procedure to allow authorization for a ballot where multiple votes for a specific position are permitted.
- Shown how Identicon voter validation would ensure an anonymous and verifiable election process.

Our voting tool allows for all of the current features of a traditional election process, but also provides key benefits:

- A voter has the ability to vote from any polling station
- No person is capable of voting more than once
- Votes are determinately, but anonymously stored to prevent tampering
- Immediate voting results upon the closure of the polls

Finished Product

```ruby
def grant_voting_right captain_address, captain_password, voter_address
  transaction = {
    to: contract_address,
    from: captain_address
  }

  target_caller = ""

  possible_callers = get_method_signatures
  possible_callers.each do |caller|
    target_caller = caller if caller == VOTE_GRANT_METHOD_NAME
  end

  method_id = get_method_id target_caller

  transaction[:data] = method_id + voter_address.without_ethereum_header

  $personal.send_transaction! transaction, captain_password
end
```

```ruby
def cast_vote voter_address, password, selection
  transaction = {
    to: contract_address,
    from: voter_address,
    gasPrice: "0x9"
  }

  target_caller = ""

  possible_callers = get_method_signatures
  possible_callers.each do |caller|
    target_caller = caller if caller == VOTE_METHOD_NAME
  end

  method_id = get_method_id target_caller

  puts "Selection is #{selection} (that's #{selection.to_ethereum_string.without_ethereum_header})."

  transaction[:data] = method_id + selection.to_ethereum_string.without_ethereum_header

  $personal.send_transaction! transaction, password
end
```

```ruby
def get_tally selection
  transaction = {
    to: contract_address
  }

  target_caller = ""

  possible_callers = get_method_signatures
  possible_callers.each do |caller|
    target_caller = caller if caller == VOTE_TALLY_METHOD_NAME
  end

  method_id = get_method_id target_caller

  formatted_selection = selection.to_ethereum_string.without_ethereum_header
  transaction[:data] = method_id + formatted_selection

  $eth.call(transaction).from_ethereum_int
end
```

```
function vote(bytes32 writin) public {
    bool isWrite = false;
    //bytes32 writein = stringToBytes32(writin);
    uint pnum = 0;
    Voter storage sender = voters[msg.sender];
      for(uint v = 0; v < sender.vote.length; v++){
       require(writin != proposals[v].name);
       require(writin != proposals[sender.vote[v]].name);
    }
    require(!sender.voted);
    for(uint p = 0; p < proposals.length; p++){
        if(proposals[p].name==writin){
            isWrite = true;
            pnum = p;
            break;
        }
    }
```

Practical Implementation Details

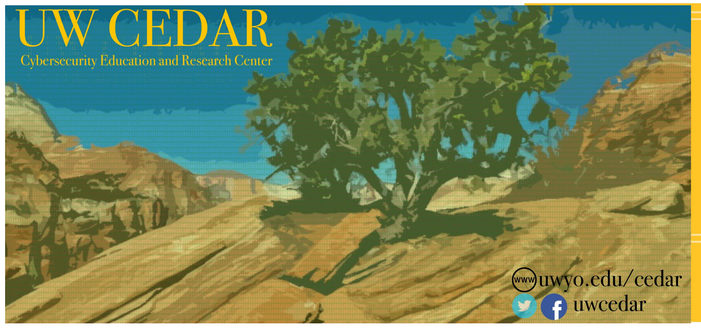
To improve our system further, we researched a way to ensure voter anonymity that allows a voter to be able to verify that their vote was cast accurately using a uniquely individual Identicon that traces a specific individuals vote in the election. This Identicon would allow the user to see their vote without their information being attached to this record.

Additionally, we provided a proposal for the Wyoming Government to update their election guidelines to include measures for an electronic system, including a contingency plan for power outages that could disrupt electronic voting services.

Conclusions & Future Work

We were able to implement a complete voting tool for modern elections, and provide a secure method for storing and counting votes in a way that has never before been implemented. Our voting tool completely encompasses the Wyoming Government's standards for voting procedures, and provides an unprecedented benchmark in voter security.

Through our implementation of an electronic voting system using blockchain technologies, we will be providing a tool for the Wyoming government to revolutionize the election process in the State so that its citizens can be confident in a secure, robust election system. Wyoming, being the current leader in blockchain technologies will be able to set an example for the rest of the nation to similarly reevaluate the election process.



UNIVERSITY OF WYOMING

Acknowledgements: All of our code was built with Solidity, Parity, and Ruby on Rails. Verum Electus is built on the Ethereum smart contract specification.

This poster made possible by: UWYO College of Engineering & Applied Science, UWYO Computer Science Department, and UWYO CEDAR.