Blockchain Voting System

Group 1

Problem Statement

Voting needs to be a secure and accurate process. Voters need to be validated and votes should be counted correctly. The current process is prone to human errors such as mistakes in counting votes, not being able to correctly validate a voter, and incorrect submissions by voters. Create a blockchain voting system that addresses these issues.

Existing Solutions

Existing Solutions

There are many projects that have been created for an online voting system. The following projects implemented the basic requirements and some more to accomplish a viable voting system. These solutions are not commercial and are uploaded to github. A couple of examples are:

- **Jawanndenn [2]:** This voting system was created in python as a libre alternative to an already existing system ‘doodle’. This system implemented Django framework for the web interface and created a simple schedule and polling system.
- **Schoolvotes [1]:** This voting system was created entirely in php and css. This is a simple voting system that entirely involves web development and is used for school elections. School elections involve many students voting for president, vice president, and more, which this system solves. This system however has a few flaws, one being that it is entirely in php, which is a dead language. That puts the system at a setback as scalability, performance and the user interface cannot be easily redesigned.

Proposed Solution

Assumptions

The system confirms the ID’s validity, and facial recognition software makes sure the person in the self-portrait is the person on the ID. Then the user is authenticated as eligible to cast a vote. This alone has its hurdles. Our project will assume we have cleared this obstacle.

System Design & Implementation

Communication & Naming

- Remote Method Invocation (RMI)
- Multicast
- Hybrid (Centralized and Decentralized)

System Design & Implementation

Coordination

- Raft Protocol

System Design & Implementation

Fault tolerance & Security

- Byzantine Fault Tolerance
- Hashing Cryptography

System Design & Implementation

Consistency & Replication

- Data-Centric Model
- Primary-Based Protocol

Architecture

The blockchain will be implemented as records in 3 separate processes hosted on different computers along with a user interface for logging in and casting a vote. The system is implemented in Java.

Demo

Evaluation of System

Usability

- Any user that exists in the database can vote
  - Pre-existing users for demonstration
- Each user can only vote once

Performance

- Latency
  - Length of time it takes a client to submit a vote
  - Response time

Security

Public-key encryption/decryption will be used to verify users through their drivers licence.

Privacy

-Each voter’s choice of candidate should be anonymous during and after the election.

Challenges

- Modifying Method Invocations for RMI
- Implementation of Multicast with RMI
- Integrating Blockchain functionality with Multicast Sender and Receiver

Conclusion and Future Work

Future work

-Scale the project

Questions?