Blockchain-Based Voting System for Secure Distributed Elections

Harvard John A. Paulson School of Engineering and Applied Sciences

Blockchain-Based Voting System for Secure Distributed Elections

Waseem Ahmad
Harvard College
waseemahmad@college.harvard.edu

Max Peng
Harvard College
mpeng@college.harvard.edu

05/04/25

Abstract

We present a prototype blockchain-based voting system that combines classical distributed-systems principles with modern Web3 tools to deliver secure, transparent, and decentralized elections. Our design builds on an in-class exploration of consensus algorithms,extending our Paxos/Raft/BFT exercises to a Proof-of-Authority (PoA) scheme,and integrates off-chain storage via IPFS alongside Ethereum-style wallet authentication. The system comprises a Next.js client for election management and vote casting, a Node.js/Express server exposing REST endpoints, a custom PoA consensus module in JavaScript, and IPFS for candidate metadata persistence. We validated our approach with end-to-end tests across multiple nodes. Our evaluation demonstrates that PoA reduces computational overhead while ensuring tamper-resistant vote ordering, IPFS guarantees data availability without prohibitive gas costs, and Web3 wallets enable strong voter authentication. This work illustrates how distributed-systems theory can be translated into a real-world application for trustworthy elections, and provides a foundation for further enhancements in scalability, privacy, and more complicated models.

The GitHub repository for this project is available at: https://github.com/waseemahmad1/cs2620_final

CS 2620: Introduction to Distributed Computing 🎓

Waseem Ahmad | Max Peng

Introduction

- Elections today still rely on physical polling and centralized back-ends.
- This model struggles with accessibility, scaling to large electorates, and transparent audits.
- Distributed systems and blockchain promise a new approach: tamper-resistant ledgers, no single point of failure, and fully verifiable results.

Motivation

- We want to apply CS 2620 principles like consensus, replication, fault tolerance to a real-world voting scenario.
- Blockchain lectures inspired us to explore this system.
- Our goal is a prototype that shows how a decentralized, secure, and transparent voting system could work in practice.

Problem Formulation

Following our proposal, we aim to design a system that:

1. Ensures security via vote integrity

Each vote, once cast, is immutable and tamper-proof through decentralized ledger storage. We plan to evaluate resilience against adversarial attempts and simulated node failures .

2. Promotes transparency while preserving anonymity

All recorded votes are publicly verifiable on the chain and through our audit mechanism, yet voter identities remain anonymous.

3. Eliminates central dependence via fault tolerance

The system does not rely on a single trusted authority for validation or storage. Instead, vote blocks are pinned to IPFS across multiple peers, and block inclusion is managed by a distributed consensus protocol rather than a central server

Our prototype mirrors real-world election systems by combining:

A web-based voting client with Web3 wallet integration

A permissioned PoA blockchain for vote transactions

IPFS for decentralized storage of metadata

A REST-based audit endpoint for independent result verification

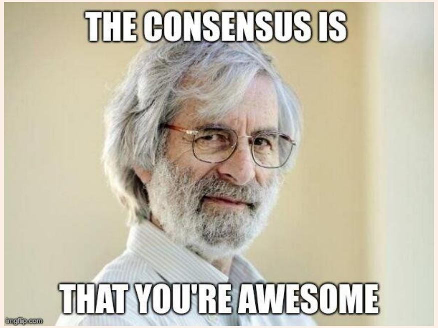
Methodology Overview

System Overview

Our voting platform integrates a Web3 client, a Node.js/Express backend, a private PoA blockchain, IPFS for off-chain storage, and a light Ethereum-style ledger for auditability.

Consensus Algorithm: Proof-of-Authority rotation for fast, permissioned block sealing	Decentralized Storage (IPFS): Pinning of candidate metadata and sealed vote blocks
Ethereum-Style Blockchain: Off-chain smart-contract ledger without on-chain mining	Audit Mechanism: /auditProof endpoint for end-to-end vote verification

Consensus Algorithm



Validators are a fixed set of known identities whose Ethereum-style wallet addresses live in config/validators.json

At each round, validators take turns (round-robin) sealing blocks of votes without any mining work

Sealing a block simply means signing the block header (parent CID + new CIDs) off-chain and submitting that signature to the ledger

IPFS (InterPlanetary File System)

Candidate metadata, ballot definitions, and each sealed block's full vote payload are serialized to JSON

These JSON files are pinned to IPFS, producing immutable content identifiers (CIDs)

The blockchain only stores CIDs, keeping on-chain data minimal while ensuring data availability and tamper resistance

Ethereum-Style Ledger

A simple smart contract exposes an `appendBlock` method that validators call off-chain

Each call records the parent CID, new CIDs array, and the validator’s anonymized signature

No proof-of-work or on-chain mining; PoA ensures fast finality with minimal gas fees

Full block history is available via JSON-RPC event logs

```
const { keccak256, toUtf8Bytes } = require('ethers').utils;

class Block {
  constructor(index, timestamp, transactions, previousHash, validator) {
    this.index = index;
    this.timestamp = timestamp;
    this.transactions = transactions;
    this.previousHash = previousHash;
    this.validator = validator;
    this.hash = this.computeHash();
  }

  computeHash() {
    const data = `${this.index}${this.timestamp}${JSON.stringify(this.transactions)}${this.previousHash}${this.validat
    return keccak256(toUtf8Bytes(data));
  }
}

module.exports = { Block };
```

Audit Mechanism

The `/auditProof` endpoint accepts an election ID and returns all recorded block headers

Server fetches contract events, retrieves each block’s JSON from IPFS, and rebuilds vote lists. Ensures transparency and allows everyone to view results

Development Log

Most work done in a 9-10 day period.

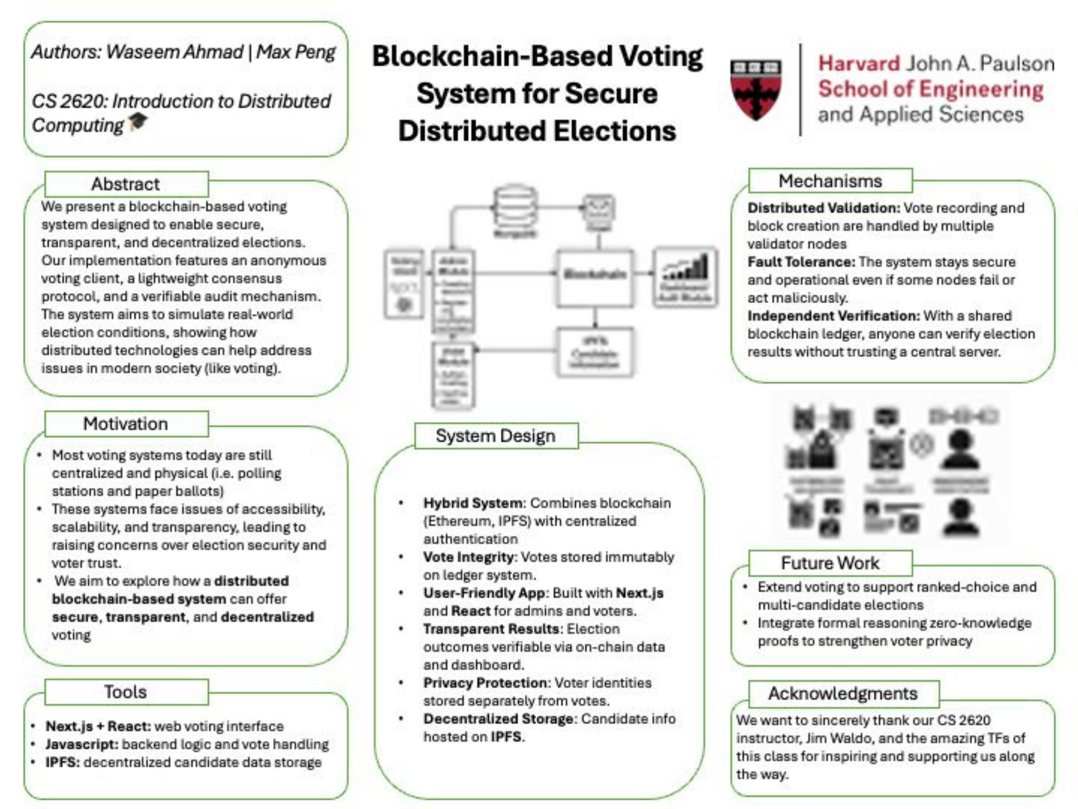
Days 1-3: Define blockchain data structures and consensus algorithm

Days 4-6: Integrate IPFS for off-chain storage, voter authentication with metamask

Days 7 - 10: Work on frontend, write out paper and presentation

Results

- General System Design
- Design Choices
- DEMO



Let’s demo our project!!!
(we may need to borrow a computer)

Thank you to the CS 2620 Professor Waldo and all the teaching staff for helping make this class an incredible experience!!!