2CSDE93- Blockchain Technology

Lab 6- To build, implement and test voting mechanism using Ethereum Blockchain. First, list the contestants on the screen and the vote they got. Whenever the user tries to vote a particular contestant, the count of the votes for the particular contestant should increase by 1. Also, the user who has already voted should be marked. Marked means “the user has already voted once and will not be allowed to vote again”.

Flow of the presentation

- ❑Traditional Voting Process
- ❑Voting Requirements
- ❑Limitations and shift towards electronic voting
- ❑E-Voting Process
- ❑Blockchain based E-Voting
- ❑The architecture of Blockchain-based E-Voting
- ❑Comparison of current blockchain voting platforms
- ❑ Challenges
- ❑Voting Smart Contract
- ❑Functions in the Smart Contract
- ❑Basic Declarations
- ❑Owners as the modifier
- ❑Function-Election name and add candidate
- ❑getNumCandidate and authorize function
- ❑Vote function and vote close function
- ❑Compiling and executing the contract

Traditional Voting Process

- It was manual and paper-based
- Majority-voting
- Not fair as the process can be made biased towards a particular party.
- Thus, posed significant threats to the democratic system.

Figure 1- Ballot-based voting Image URl- https://www.shutterstock.com/search/vote+icon

Voting Requirements

Limitations and shift towards electronic voting

- Need for anti-corruption protection while ensuring the voting process is correct.
- In comparison to manual voting, the focus has shifted towards e-voting as it increases reliability over the manual process.
- E-voting is flexible, simple to use, and scalable solution.

Limitations of E-Voting systems-

- E-voting systems are challenged due to over-authority and manipulation in details.
- It limits the fundamental fairness, privacy, and secrecy of data.
- It limits the fairness as the voting process is centralized, and thus is subjected to security and privacy based attacks
- Another important aspect is the anonymity of the voters, which is not established in centralized e-voting process.
- Trust and transparency is not present.
- The voting process is licensed by the central authority, and is controlled and monitored by the central users.
- Cannot stop the collusion among parties and the central election commission.

E-Voting Process

1. The election commission
2. E-Ballots
3. Cryptographically secured votes

Traditional Voting System

Challenges-

Trust among multiple sites.
Collusion based attacks
High overhead of security communication

Why shift towards decentralized voting?

Reduce the computational bottlenecks on central servers.
Address the issue of single-point of failure
Redundancy of data at multiple sites.

Blockchain based E-Voting

Blockchain technology offers a decentralized node for online voting or electronic voting.

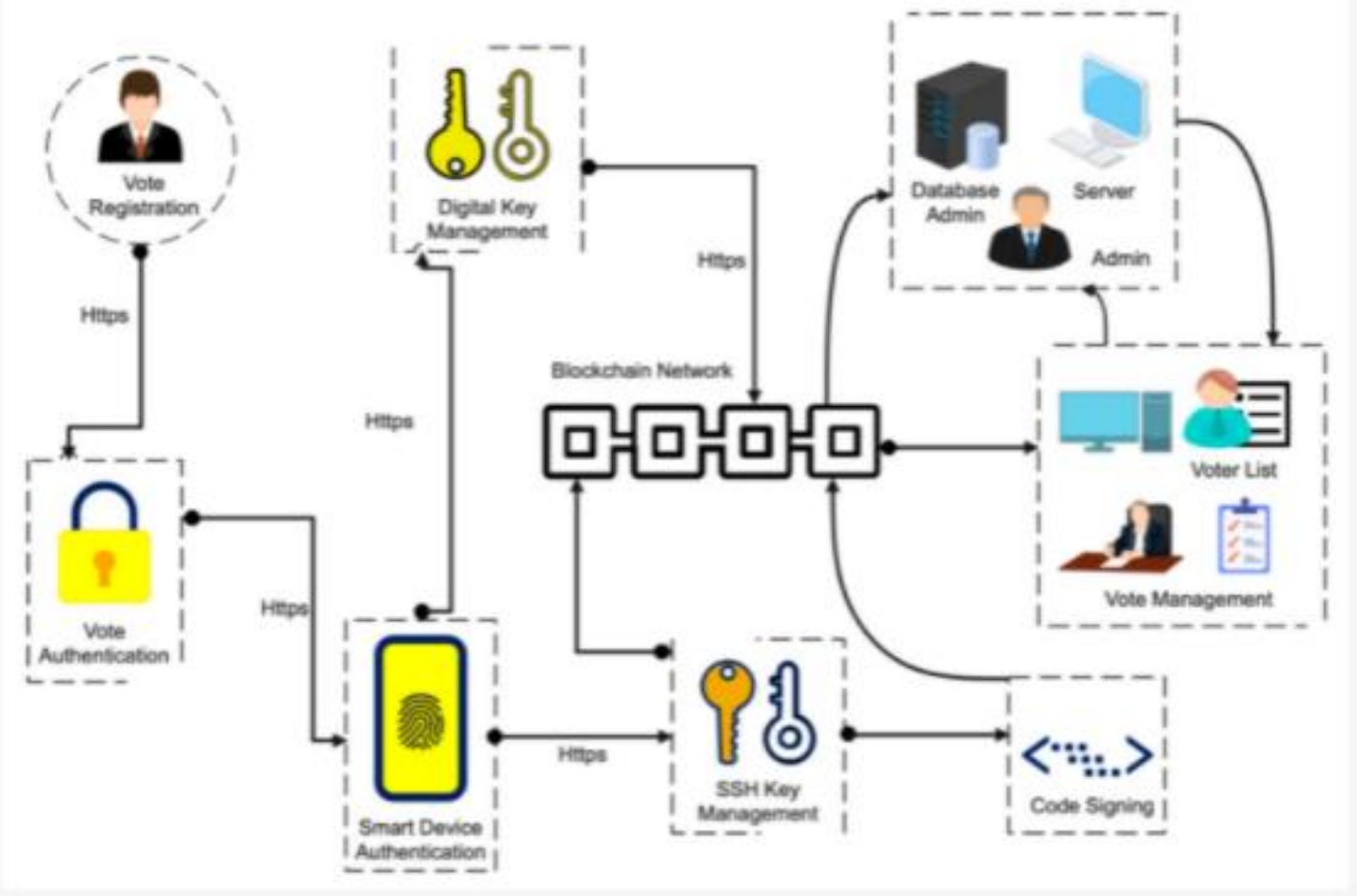
Recently distributed ledger technologies such blockchain were used to produce electronic voting systems mainly because of their end-to-end verification advantages.

Blockchain is an appealing alternative to conventional electronic voting systems with features such as decentralization, non-repudiation, and security protection.

Voting is a new phase of blockchain technology; in this area, the researchers are trying to leverage benefits such as transparency, secrecy, and nonrepudiation that are essential for voting applications.

With the usage of blockchain for electronic voting applications, efforts such as utilizing blockchain technology to secure and rectify elections have recently received much attention

The architecture of Blockchain-based E-Voting

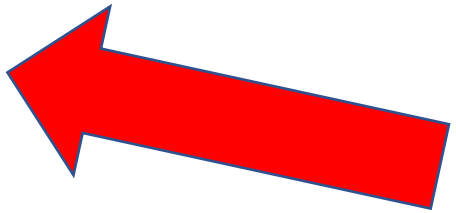


Comparison of current blockchain voting platforms

Online Voting Platforms	Framework	Language	Cryptographic Algorithm	Consensus Protocol	Main Features (Online Blockchain Voting System)							
					Audit	Anonymity	Verifiability by Voter	Integrity	Accessibility	Scalability	Accuracy/Correctness	Affordability
Follow My Vote	Bitcoin	C++/Python	ECC	PoW	✓	✓	✓	✓	✓	✗	✓	✓
Voatz	Hyperledger Fabric	Go/JavaScript	AES/GCM	PBFT	✓	✓	✓	✓	✓	✗	✓	✓
Polyas	Private/local Blockchains	NP	ECC	PET	✓	✓	✓	✓	✓	✗	✓	NA
Luxoft	Hyperledger Fabric	Go/JavaScript	ECC/ElGamal	PBFT	✓	✓	✓	✓	✓	✗	✓	✓
Polys	Ethereum	Solidity	Shamir's Secret Sharing	PoW	✓	✓	✓	✓	✓	✗	✓	✓
Agora	Bitcoin	Python	ElGamal	BFT-r	✓	✓	✓	✓	✓	✗	✓	✓

Challenges

- Preventing Impersonation.
- Preventing fraud.
- Preventing voter/vote privacy violations
- Relinquishing the uses of any intermediaries.
- Prevent coercion freeness.



A voting contract is required to cast votes and counts

Voting Smart Contract

- Candidates to contest in election
- Voter Entity
- Contract Owner
- Election Name (given by election commission)

Functions in the Smart Contract

- electionName
- addCandidate
- getNumCandidate
- Authorize
- Vote
- Voting end

Requirements-

1. A candidate is authorized and then is able to cast his vote.
2. Only one vote is casted by a candidate.
3. Once votes are casted, the owner frees the contract from memory

Basic Declarations

- Candidate and Voter

```
 struct Candidate {
        string name;
        uint voteCount;
}
struct Voter {
        bool authorized;
        bool voted;
        uint vote;
}
```

Owners as the modifier

```
address public owner;
string public electionName;
mapping(address => Voter) public voters;
Candidate[] public candidates;
uint public totalVotes;
modifier ownerOnly() {
        require(msg.sender == owner);
        _;      //remaining body of addCandidate to be executed
}
```

Function-Election name and add candidate

```
function Election(string _name) public {
        owner = msg.sender;
        electionName = _name;
}
function addCandidate(string _name) ownerOnly public {
        candidates.push(Candidate(_name,0));
}
```

getNumCandidate and authorize function

```
function getNumCandidate()
public view returns(uint) {
        return candidates.length;
}

function authorize(address _person) ownerOnly public {
        voters[_person].authorized = true;
}
```

Vote function and vote close function

```
function vote(uint _voteIndex) public {
        require(!voters[msg.sender].voted);
        require(voters[msg.sender].authorized);
        voters[msg.sender].vote = _voteIndex;
        voters[msg.sender].voted = true;
        candidates[_voteIndex].voteCount += 1;

        totalVotes += 1;
}
function end() ownerOnly public {
        selfdestruct(owner);
}
```

Compiling and executing the contract