Decentralized Voting System

Problem Description

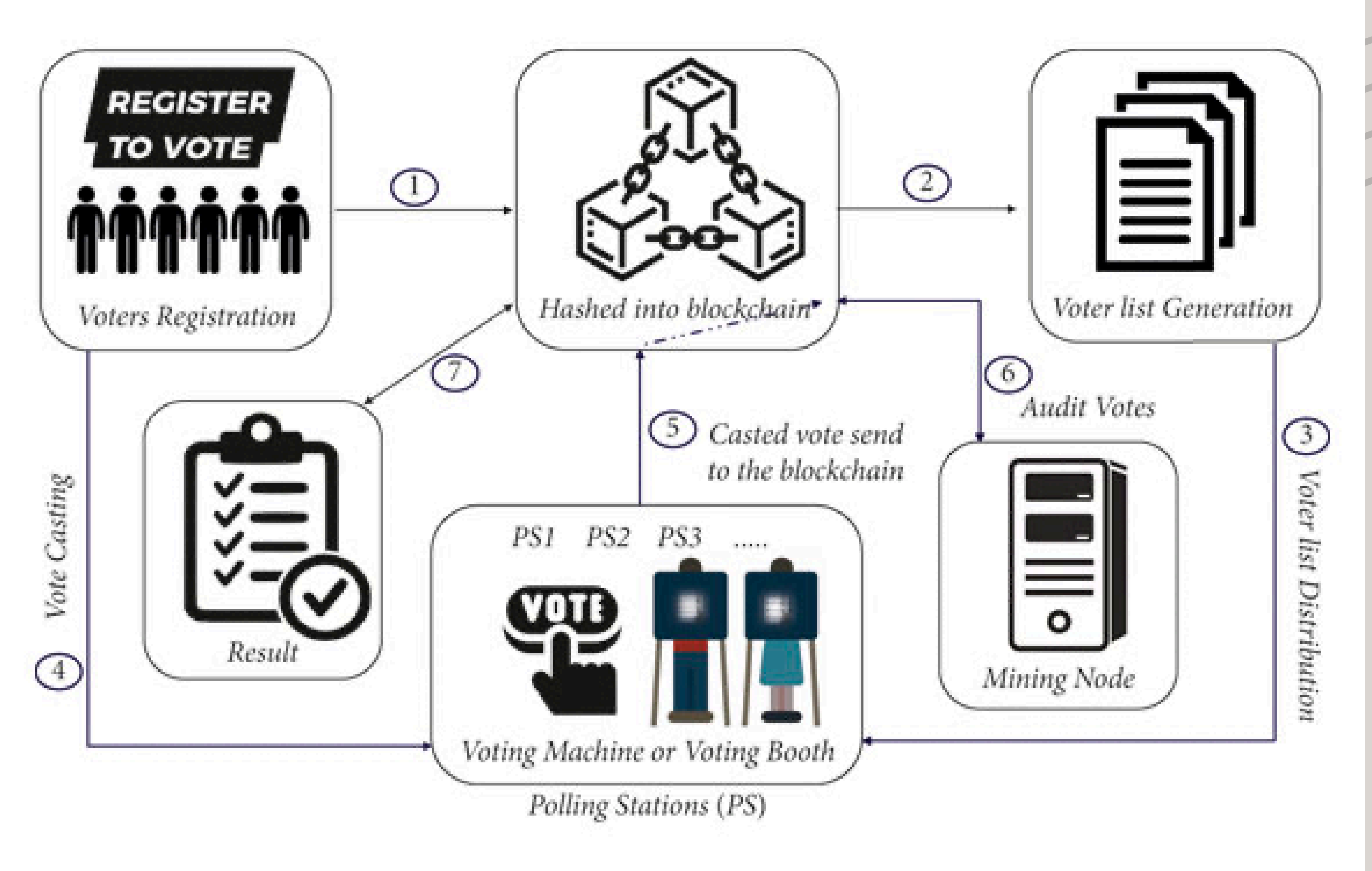
Traditional voting systems are often plagued by issues such as:

- Centralized control, which can lead to manipulation and fraud.
- Lack of transparency, making it difficult to verify the integrity of the results.
- Vulnerability to cyber-attacks, affecting voter data and election outcomes.

The need for a secure, transparent, and immutable system to ensure trust in the democratic process.

Solution Overview

- Implementing a Decentralized Voting System using blockchain technology.
- Each vote is a secure transaction recorded on the blockchain.
- Decentralized nature eliminates the possibility of central authority manipulation.
- Immutability guarantees an unforgeable record, fostering confidence in the election's integrity.



Implemented Functionalities

01

Voter Registration: Only registered voters can vote.

02

Voting: Registered voters can cast their vote for a valid candidate.

03

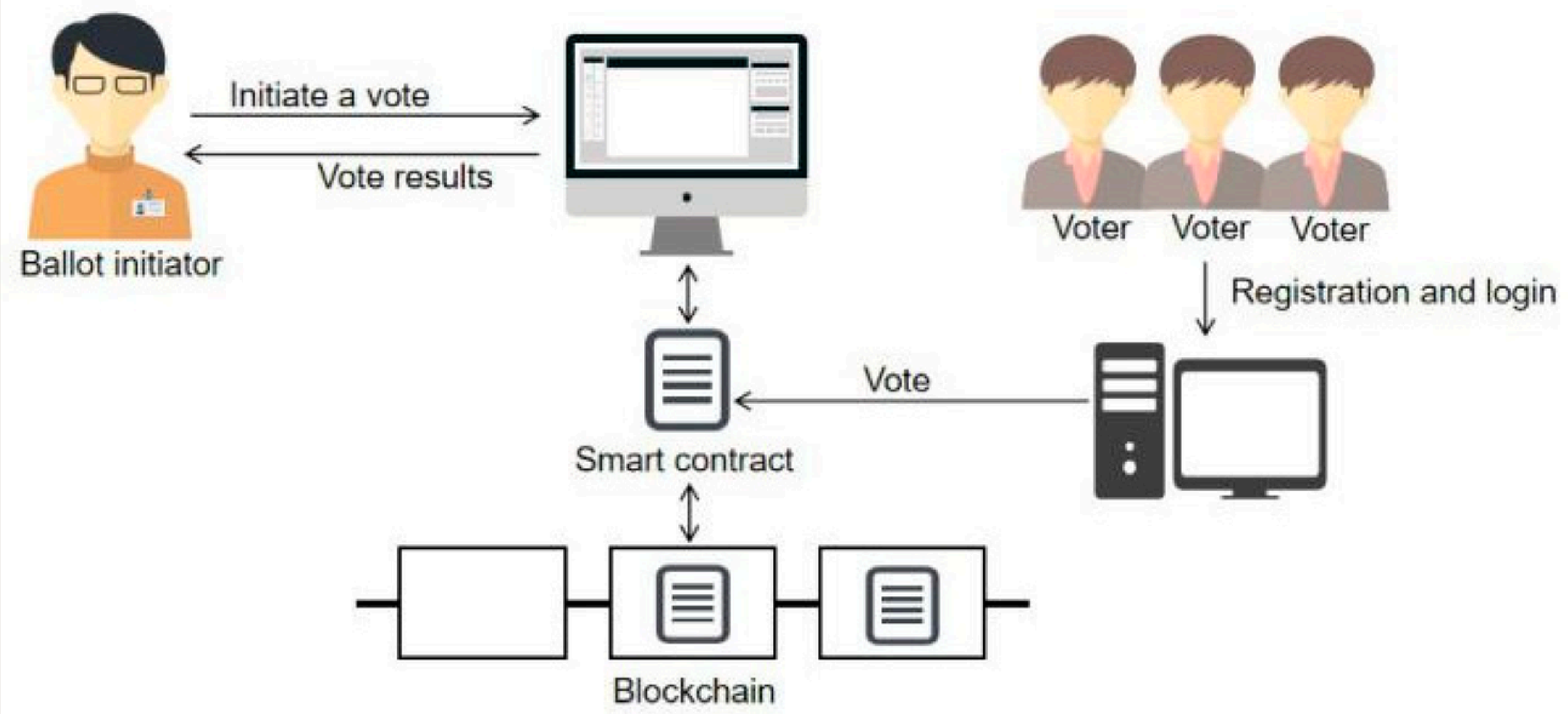
Vote Counting: Transparent and real-time count of votes received by each candidate.

Application Architecture

- Frontend: User interface for voter registration and voting (not implemented in this solution).
- Backend: Smart contract deployed on the Ethereum blockchain.
- Data Storage: Votes and voter registration data stored on the blockchain.
- Design Decisions:
  - Decentralized storage to ensure transparency and security.
  - Smart contract for automated and trustless vote counting.

Smart Contract Overview

- The smart contract is written in Solidity.
- It handles voter registration, vote casting, and vote tallying.
- Ensures only registered voters can vote and only once.
- Maintains a list of valid candidates and tallies votes for them.



Voter Registration

- Only the contract owner can register voters.
- Ensures the voter is not already registered before adding them.

```
function registerVoter(address voter) public onlyOwner {
    require(!voters[voter], "Voter is already registered.");
    voters[voter] = true;
}
```

Voting

- Checks if the voter is registered.
- Checks if the candidate is valid.
- Records the vote and prevents the voter from voting again.

```
function vote(bytes32 candidate) public {
    require(voters[msg.sender], "Only registered voter can vote.");
    require(validCandidate(candidate), "No such candidate.");
    votesReceived[candidate] += 1;
    voters[msg.sender] = false;
}
```

Vote Counting

- Ensures the candidate is valid before returning the vaote count.
- Provides a transparent way to check vote totals.

```
function totalVotesFor(bytes32 candidate) view public returns (uint256) {
    require(validCandidate(candidate), "No such candidate.");
    return votesReceived[candidate];
}
```

Conclusion

Enhanced Security:

- Blockchain technology ensures that each vote is securely recorded as a transaction. This eliminates the risks associated with traditional voting systems, such as tampering and hacking.
- The decentralized nature of the system means there is no single point of failure or control, making it more resistant to attacks.

Transparency:

- The use of a public ledger means that the voting process is transparent. Any participant can independently verify the integrity of the vote counts, fostering trust among voters.
- Voters can see their vote being recorded, and anyone can audit the entire voting process to ensure accuracy and fairness.

Conclusion

Immutability:

- Once a vote is cast and recorded on the blockchain, it cannot be altered or deleted. This immutability guarantees that the vote count remains accurate and unforgeable, preserving the integrity of the election.

Trust in the Electoral Process:

- By removing the possibility of central authority manipulation and ensuring the security and transparency of votes, the system instills confidence in the electoral process.
- Voters are more likely to participate in elections when they trust that their vote will be counted accurately and fairly.

Hognogi Ana-Maria Cristina

Thank You

Iakab Alex-Edward