Objective

The goal of this assignment is to implement a decentralized voting system using the zkSync Layer 2 testnet. This system must ensure voter privacy using zk-proof verification, while also maintaining the integrity and fairness of the voting process. We will write a Solidity smart contract, deploy it on zkSync Testnet, and test the contract's functionality using Remix IDE.

Requirements

You are required to build a voting system with the following features:

1. **Election Creation:**
   - Administrators should be able to create elections by specifying the name of the election, start and end times, and a list of candidates.
   - Each election must have a unique identifier (e.g., `electionId`).
2. **Vote Submission:**
   - Voters must submit their votes as cryptographic proof hashes (`proofHash`).
   - The contract must ensure that no proof hash is used more than once.
   - Votes must be recorded against the chosen candidate.
3. **Result Declaration:**
   - Only the administrator who created the election should be able to declare the results after the election has ended.
   - The contract must identify the winner by tallying votes for all candidates.
4. **Information Retrieval:**
   - Anyone should be able to query the list of candidates for a given election.
   - Anyone should be able to query the total votes for any candidate.
   - The contract must allow querying whether a proof hash has already been used.

Assignment Steps

1. Write the Smart Contract

- Use the Solidity programming language to write the voting system contract.
- Implement the following functionalities:
  - Election creation (`createElection` function).
  - Vote submission (`submitVote` function).
  - Result declaration (`declareResults` function).
  - Candidate and vote queries (`getCandidates`, `getVotes`, and `isProofUsed` functions).

2. Deploy the Contract on zkSync Testnet

3. Test the Smart Contract

- Write test cases for the following scenarios:
    1. Create an election with a list of candidates.
    2. Submit votes for various candidates using different proof hashes.
    3. Ensure that duplicate proof hashes are rejected.
    4. Declare results and verify the winner.
    5. Query the total votes for each candidate.
- Perform all tests on the zkSync Testnet.

Evaluation Criteria

Your submission will be evaluated based on the following:

1. **Functionality:** Does the contract implement all the required features correctly?
2. **Code Quality:** Is your code clean, modular, and well-documented?
3. **Gas Optimization:** Have you minimized gas costs for key operations (e.g., vote submission)?
4. **Test Coverage:** Have you covered all edge cases in your testing?
5. **Deployment:** Has the contract been successfully deployed and tested on zkSync Testnet?

Submission Guidelines

1. Submit the following files in a zipped folder:
    - `VotingSystem.sol`: Your Solidity smart contract file.
    - `testCases.md`: A markdown file listing all test cases you performed and their results.
    - `deploymentLink.txt`: A text file containing the deployed contract address on zkSync Testnet.
2. Include a brief report (`report.pdf`) describing your approach, challenges faced, and how they were overcome.
3. Please upload your .sol file to the elearn platform.
   Deadline:
   (December 16, 2024), (Azar 26, 1403), (Jumada al-Thaniyah, 14, 1446)

Helpful Resources

- Remix IDE: `https://remix.ethereum.org/`
- zkSync Documentation: `https://docs.zksync.io/`
- zkSync Faucet: `https://docs.zksync.io/build/tooling/network-faucets.html`
- ZKsync Bridge: `https://portal.zksync.io/bridge/`
- Chainlist: `https://chainlist.org/`
- Solidity Documentation: `https://docs.soliditylang.org/`

Bonus Challenge (Optional)

Enhance the privacy of the voting system by fully integrating zk-proofs for vote validation. Use tools like **Circom** or **snark.js** to generate zk-proofs off-chain and verify them within the smart contract.

If you have any questions or need assistance with the tasks, feel free to reach out.

Best regards,

Reza Nematpour

https://www.youtube.com/watch?v=MGemhK9t44Q