USI Winter School 2024
Writing Smart Contracts (WSC)
Final Project

Kristina Odermatt
odermk@usi.ch

(6) Voting

A club has a membership list with the public keys of all members.
Write a smart contract that allows members to vote for one of three candidates (Rossi, Smith or Meier).
Extension: (1) Ensure that voting is only possible in a certain time frame.

Table of Contents

1 Project Description

The objective of this project is to explore the utilization of blockchain technology in facilitating a secure and transparent voting mechanism for internal club elections, moving beyond its conventional financial applications. It aims to provide an overview of how blockchain technology can be applied to democratize internal election processes, ensuring that they are conducted with security and fairness.

The approach used for this project includes:

- Developing a smart contract: a smart contract on the Algorand blockchain is created and deployed to administer the voting process, allowing club members to cast votes for one of three predefined candidates: Rossi, Smith or Meier.

- Incorporating timeframe restrictions: a feature of the smart contract is its ability to restrict voting to a specific timeframe. This functionality is designed to augment the integrity of the voting process, ensuring that all electoral activities occur within a pre-established period.

- Ensuring single vote per member: in order to ensure that the fairness principle is maintained, the smart contract is designed to ensure that every member has the right to one vote.

- Streamlining the election process: the project outlines the complete election process, starting with member opt-in and continuing through voting and the final stages of closing out and optionally deleting the smart contract.

2 Smart Contract Address

Voting Smart Contract App ID: 640731779

Explorer Link: https://testnet.explorer.perawallet.app/application/640731779

3 Notebooks

Each notebook serves a specific purpose in the development, deployment and management of the smart contract voting system, designed with security, transparency and user accessibility in mind.

3.1 "*0_Accounts_project*"

Purpose:

This notebook is designed to set up the foundational elements of the voting system by creating Algorand accounts for the club and its members. It outlines the process of generating accounts, securing private keys and preparing for interaction with the Algorand blockchain.

Key functionalities:

- Account generation: automatically generates Algorand accounts for members and the club, ensuring each entity has a unique identifier within the blockchain.
- Credential management: demonstrates secure handling of private keys and mnemonic phrases, crucial for account security and recovery.
- JSON storage: details the process of storing account details in a JSON file for easy access and management within the project's scope.

Design decisions:
The decision to use JSON for credential storage facilitates easy integration with other notebooks and simplifies credential management across the project lifecycle. Automating account creation streamlines the setup process for users, ensuring a smooth start to the voting process.

3.2 "*01_Create_SC*"

Purpose:
This notebook focuses on the creation and deployment of the smart contract on the Algorand testnet, laying the groundwork for the voting system.

Key functionalities:

- Smart contract compilation: guides through the process of compiling the smart contract written in PyTeal to TEAL, the Algorand Smart Contract language.
- Deployment: illustrates the steps required to deploy the smart contract on the Algorand testnet, making it accessible to club members for voting.
- Voting time frame: incorporates logic to restrict voting to a predefined time frame, enhancing the contract's security and integrity.

Design decisions:
Implementing a voting time frame directly within the smart contract ensures that votes can only be cast during the designated period, a critical feature for maintaining the election's fairness. For this project 200 rounds were added as the voting period, which shoud be approximately 15 minutes.

3.3 "*02.1_Use_SC_Voters*"

Purpose:
This notebook provides steps for voters on how to interact with the smart contract, including opting in, casting votes and verifying vote status.

Key functionalities:

- Opt-In process: detailed steps for voters to opt into the smart contract, a necessary step before casting votes.
- Casting votes: instructions for voters to cast their votes securely through the smart contract.

- Vote verification: demonstrates how voters can verify their vote has been recorded correctly in the blockchain.

Design decisions:
Initialization of the total number of club members within the smart contract's global state to a fixed number (15). This approach is chosen to:

- Ensure clarity and fairness: setting a fixed number of members at the outset provides clarity on the scope of the election and ensures that the voting process is fair and transparent. It establishes clearly who is eligible to participate.

- Simplify management: by starting with a fixed number of members, the smart contract simplifies the management of the voting process. It eliminates the need for dynamic adjustments to the member count, making it straightforward to track participation rates and ensure that only authorized members cast votes.

- Maintain election integrity: the fixed member count contributes to the integrity of the election by preventing unauthorized additions to the electorate during the voting period.

3.4 "*02.2_Use_SC_Club*"

Purpose:
This notebook guides the club through managing the smart contract, including monitoring voting progress and closing out the contract post-election.

Key functionalities:

- Monitoring votes: provides methods for the club to monitor vote counts and participant engagement in real-time.
- Closing Out/Deleting the app: details the process for the club to close out from the smart contract, ensuring a clean conclusion to the voting event.

3.5 "*03_Testing_SC*"

Purpose:
Testing the smart contract against various scenarios to ensure its robustness and security, including attempts to vote for non-existing candidates, double voting, and voting outside the permitted timeframe.

Key functionalities:

- Error simulation: simulates common voting errors and unauthorized actions to test the smart contract's response mechanisms.
- Security checks: validates the smart contract's security features, ensuring that only authorized votes are counted

4 How to use the Smart Contract for end users

By adhering to the following credential management and funding guidelines, participants can ensure a smooth and secure experience with the smart contract voting system on the Algorand blockchain.

Firstly, to interact with the Algorand blockchain and the smart contract voting system, participants will need to manage:

- Algorand account addresses: each club member and the club itself must have an Algorand account. The account address is used to identify participants within the blockchain and to execute transactions, such as voting or smart contract management actions.

- Private keys and mnemonic phrases: the security of an Algorand account is maintained through private keys, which are often backed up as mnemonic phrases. It is vital that these are stored securely and not shared publicly, as they grant full access to the associated Algorand account and its assets.

- Smart Contract application ID: after deploying the smart contract, the club will receive an application ID unique to the smart contract. This ID is required for all transactions related to the voting process, including opting in, voting, and administrative actions by the club.

Participants must ensure their credentials are securely managed and readily accessible when interacting with the smart contract.

Moreover, interacting with the Algorand blockchain incurs transaction fees and requires maintaining a minimum balance in each account to remain active:

- Transaction fees: every transaction on the Algorand network, including voting, opting in, and smart contract management actions, incurs a nominal fee (typically 0.001 ALGO). Participants must ensure their accounts are funded to cover these fees.

- Minimum balance: Algorand accounts must maintain a minimum balance. For basic accounts interacting with a single Smart Contract, the minimum balance is usually 0.1 ALGO, but participants should be aware of this requirement to avoid disruptions.

- Funding for deployment and management: The club's account, responsible for deploying and managing the smart contract, will require additional funding. Deployment incurs a one-time fee, and managing the smart contract (e.g., closing out or deleting it) will also require transaction fees.

Lastly, it is recommended to:

- Secure storage: participants should use secure methods to store their private keys and mnemonic phrases, such as encrypted digital storage or physical safes for paper backups.

- Initial funding: It is recommended that participants initially fund their accounts with a slightly higher amount than the bare minimum to accommodate multiple transactions and any unforeseen requirements.

- Monitor balances: Participants should regularly monitor their account balances to ensure they have sufficient funds for their intended actions, especially during active voting periods.

4.1 For the Club (Organizer):

The club, acting as the organizer and administrator, uses two notebooks: "*01_Create_SC*" for setting up the smart contract and "*02.2_Use_SC_Club*" for managing the voting process.

Steps for the Club:

1) Create and deploy Smart Contract (Notebook: "*01_Create_SC*"):
   - The club begins by creating the smart contract using PyTeal, compiling it into TEAL, and then deploying it on the Algorand testnet. This process establishes the foundation for the voting system, including setting the voting period and candidates.

   - The deployment script within the notebook guides the club through generating the deployment transaction, signing it with the club's account, and submitting it to the Algorand network. Upon successful deployment, the smart contract's application ID is generated.

2) Opt-in to the Smart Contract (Notebook: "*02.2_Use_SC_Club*"):
   - Although the club is the creator of the smart contract, it must also opt-in to perform certain administrative functions and monitor the voting process.

3) Monitor voting progress:
   - Throughout the voting period, the club can monitor participation, track the number of votes each candidate receives and ensure the voting process's integrity using the "*02.2_Use_SC_Club*" notebook.

   - The notebook provides scripts for querying the smart contract's global state, offering real-time insights into the voting process.

4) Close Out and deleting the Smart Contract (optional):
   - After the voting period ends and results are tallied, the club can close out the smart contract, ending the election. This step involves executing a close-out transaction from the club's account.

- Optionally, the club can delete the smart contract to remove it from the blockchain, ensuring that the election is concluded and the data is finalized. This action is performed by submitting a delete transaction through the "*02.2_Use_SC_Club*" notebook.

4.2 For Voters (Members):

Members primarily interact with the smart contract using the notebook "*02.1_Use_SC_Voters*." This notebook guides members through the entire process of participating in the voting, from opting in to the smart contract to casting their votes and verifying their participation.

Steps for voters:

1) Opt-in to vote:
   - Members start by opting into the smart contract to participate in the voting. This step is necessary for setting up the member's account to interact with the smart contract, allowing the system to record their vote.

   - To opt-in, members execute the opt-in transaction script within the notebook. This process involves generating and signing a transaction that is then sent to the Algorand network, officially registering the member as a participant in the vote.

2) Cast vote:
   - Once opted in, members can cast their vote for one of the candidates. The notebook provides a script where members select their candidate of choice and generate a voting transaction.

   - The script ensures that the vote is cast within the designated voting period and that each member can only vote once, aligning with the smart contract's logic.

3) Verify vote:
   - After voting, members can verify that their vote has been correctly recorded by the smart contract. This verification process involves querying the smart contract's state to confirm the member's vote.

   - This step is crucial for ensuring transparency and trust in the voting process, allowing members to confirm their participation and the integrity of their vote.

5 Project Credentials and Funding Requirements

5.1 Project Credentials

To facilitate an efficient setup process for our Algorand blockchain-based voting system, I have streamlined the creation of accounts and the management of their credentials. This is accomplished through the initial notebook "*0_Accounts_Project*."

- The notebook automates the generation of Algorand accounts for both the club and its members. By executing the scripts within, users can quickly create the necessary accounts without manual intervention.

- Secure credential storage: upon creation, the account credentials, including public keys, private keys and mnemonic phrases, are automatically stored in "*credentials_project*" file. This file serves as a central repository for managing account access throughout the project, ensuring that credentials are readily accessible when needed for blockchain transactions.

5.2 Funding Accounts on the Algorand Testnet

For the purposes of this project, we utilize the Algorand testnet, which offers a convenient way to fund accounts without real-world costs.

Funding process:

- Testnet Dispenser: in the "*01_Create_SC*" notebook, I have included instructions and a markdown link to the Algorand Testnet Dispenser (https://dispenser.testnet.aws.algodev.network), a tool provided by the Algorand community.

- Dispensing ALGOs: this tool allows users to dispense 5 ALGOs to their testnet accounts at a time. These funds are necessary to cover the transaction fees incurred during smart contract deployment, voting and other blockchain interactions within the project.

- Each member account and the Club account must be funded.

6 References

WSC Notebooks provided by Prof. Peter Gruber (peter.gruber@usi.ch).