Second-term project

TX and Smart Contract

Introduction

Instead of reading papers, we are now going to interact with blockchain to understand it better. In this project, we are going to have fun with Ethereum smart contract.

Project Goal: Students will know better how smart contract and transactions in Ethereum work.

General

PJ2 smart contract address:
0xC820cBdc60c879cB73Cdd895e7e89E796f6C6C16

Enviroment : **Ropsten Ethereum Testnet**

PJ2 related files: https://github.com/yenchihliao/BlockchainIntroduction

There are 3 problem and 1 bonus in this homework

Problem1

Send a transaction to to the function, “Problem1” in PJ2.sol with any tools you like.

- The input will be your **studentID**. e.g. “b00902000”
- You need to pay at least 3 gwei to this contract

Problem2

Program for problem2 include:

- Send a transaction to to the function, “Problem2” in PJ2.sol
- The input includes your **studentID** and the **hexdigest of sha256 hash** of your program
- You need to pay at least 3 gwei to this contract

Hint : You can’t hard code the hash value into your program yet still be able to send the hash value of your program as transaction input.

Spec.

1. First alphabet in input of studentID should **not** be capitalized.
2. To calculate the hash value of your program, please use the checkFile.py here.

   ```
   $python checkFile.py [filename]
   ```

3. For contract that you need to deploy in problem3, you only need to implement one public function named **studentID** that takes no input and outputs your studentID.
4. For bonus, please trace the PJ2.sol to find it. Do not use brute-force search(all 251 possibilities)! Please try to guess within a range. You won't get bonus point if you used brute force search.

Problem3

Step1: Program for problem3 include

- Write a program that deploys the contract(will be described in spec.)

Step2: Send a transaction to to the function, “Problem3” in PJ2.sol.

- The input of PJ2.Problem3 includes your **studentID**, your **program hash(sha256)** for problem3 and **address of your deployed contract**

Problems

The contract PJ2.sol is on Ropsten Ethereum testnet:

1. Send a transaction to PJ2.Problem1 with your ID as input with at least 3 gwei.
2. Send a transaction to PJ2.Problem2 by program you write. The input includes your studentID and the hexdigest of sha256 hash of your program with at least 3 gwei.
3. Write a program that deploys the contract(will be described in spec.) described in next page and send transaction to PJ2.Problem3 afterward. The input of PJ2.Problem3 includes your studentID, your program hash for problem3 and address of your deployed contract.
4. Bonus is hidden in PJ source code.

Submission

- deadline: 11/29 23:59:59
- Upload [studentID].zip to Ceiba including
  - The program that solves problem2, problem2.[py/js/...] that matches the hash value in the transaction to problem2.
  - The program that solves problem3, problem3.[py/js/...] that matches the hash value in the transaction to problem3.
  - The smart contract you deployed in problem3, problem3.[sol/vy/...].
  - Write a report.txt that describes how you solve bonus question and any other feedback.

Some recommandations

1. Although you can use the solution of problem2 to solve problem1, you might want to try out some sophisticated tool to send transactions such as MetaMask with Remix.
2. It’s convenient to get Ropsten testnet coin by using the faucet if you are using MetaMask. (remember to switch to reposten testnet instead of mainnet)
3. When you are implementing the smart contract, you can test the correctness of your code on Remix with MetaMask with ease.
4. If you don’t know how to write a smart contract, here’s a solidity tutorial.
5. To write programs that interact with chains, the package, **web3**, is usually the choice.

Grading

There are 30(+20 bonus) points automatically judged by PJ.sol. You can see that part online directly. The rest 70 points will be judged by TA.

- 5% problem1 (automatically sumed up in PJ.sol)
- 10% problem2 submission (automatically sumed up in PJ.sol)
- 15% problem3 submission (automatically sumed up in PJ.sol)
- 20% problem2 correctness
- 20% problem3 function correctness
- 30% problem3 contract correctness
- 20% Bonus (automatically sumed up in PJ.sol)