zku.one application

What is a smart contract? How are they deployed? You should be able to describe how a smart contract is deployed and the necessary steps.

- A smart contract is a piece of code that serves the sets of instructions and conventions in the way the specific accounts on the blockchain should operate. The code for smart contract is required to be compiled and deployed using software interfaces like hardhat (for example in Ethereum). Smart contract is expected to legally bind molre than two parties working on blockchain network using credible signatures with cryptographic technology. Assuming that using Ethereum blockchain (or something built upon EVM like Polygon), one is expected to take the following steps:
    - Setting up RPC server - doing it for your own OR using Node services (e.g. Infura, Alchemy)
    - Setup utility development frameworks that help to connect to Node for deploying/compiling contracts. I have used hardhat, after trying to compare between hardhat and truffle in Ethereum development.
    - Writing the contract with Solidity.
    - Test the contract using testing framework like jests
    - Compile, Deploy (to 0x account as a recipient) to the mainnet instance.
    - Verify the contract (e.g. Etherscan)
    - Interact with the contract from frontend UI using (e.g. web3.js, etheres.js)
- The contract will subsequently be assigned a unique address for interacting with its code.
- Be careful to prepare gas fees. it costs a lot when deploying to mainnet network :)

What is gas? Why is gas optimization such a big focus when building smart contracts?

- Gas fees is known as transaction fees. It is required to cover the expense of certifying each networks. We all know that the type of transaction varies, for example, one is to update states while some requires the bunch of computational but complex logics to complete the verfication process. The certain amounts of transactions are allowed to be confirmed and attached to each blocks in given time, whereby traction-gaining networks might in the rush of a number of increasing transactions. Thus, protocols like Ethereum have built the system that calculates the computational effort in running EVM code on chain. It means that the each of smart contract functions has its own costs to run based on opcodes of Ethereum Virtual Machine(EVM). The sender is allowed to pay extra gas costs to be priortized. The gas optimization is critical for everyday users since the user has incentivized to use programs using the code when the costs of it is economically efficient (than other competitors). When we can utilize the same features for much cheaper price, we would entertain the code without exposuring secruity risks.

What is a hash? Why do people use hashing to hide information?

- A hash is a encrypted string with unique but having fixed length. The string is easily created from any other string or data, using various methods involved. The result hash string is difficult to be deciphered in backwards without additional personal keys to encrypt the string. To illustrate the example, let us say you have "I love blockchain" message. It could be encrypted into a hash like "84228b032ad69ca54729156f864eda73421e493af5723073695a15fc6fa25293" in SHA256 algorithm. While you are not able to know what the message is, however, you can confirm that the message to those who have the identical hash string was the same.
- Blockchain technology uses the hash to save the storage data, by hashing the data in each block and appending it to the hash of previous one. The newly created hash reflects the afore-created all transactions in the block.

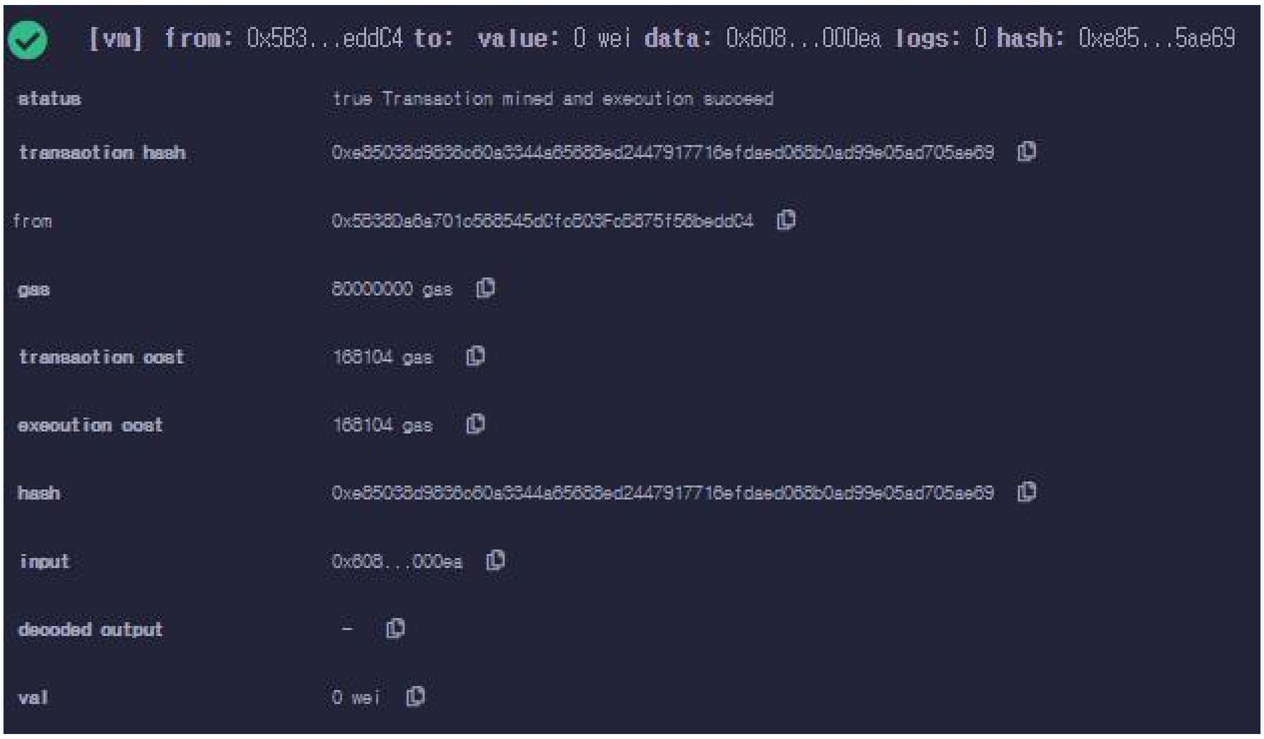
How would you prove to a colorblind person that two different colored objects are actually different colored? You could check out Avi Wigderson talk about a similar problem here.

- Allow the colorblind individual to hide items behind one's back. Then, at random, ask the individual to disclose one of the things and tell her what color it is. Then ask the individual to hide the thing behind their back and pick an object for you to identify at random with a replacement. You can be sure you're mentioning a different color every time she swaps the objects if you perform this method enough times.

You sure you're solid with Solidity?

1. I wrote a super simple "Hello World" smart contract on here: Link (https://github.com/jypthemiracle/ZKU-background-assignment/blob/master/contracts/HelloWorld.sol)

- The proof of deployment is here on Remix.

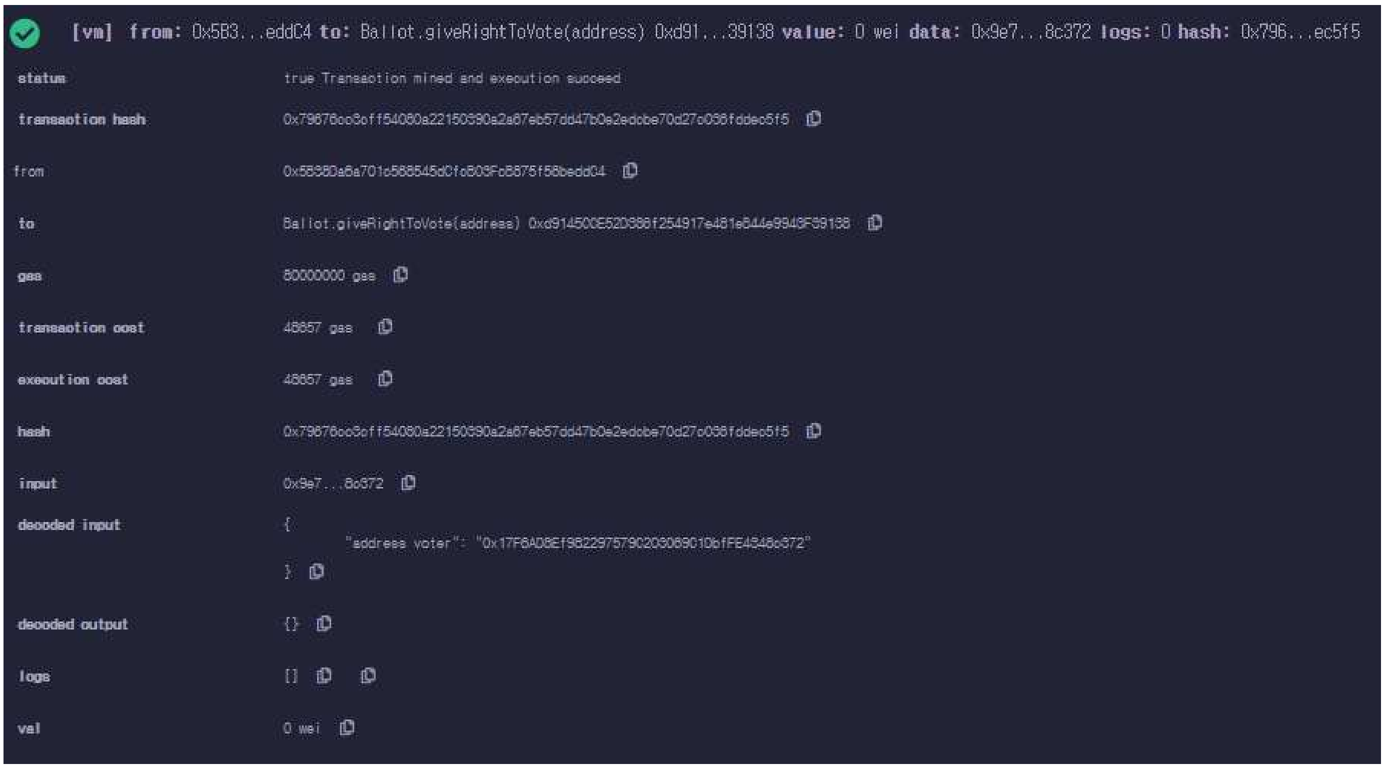


```
[vm] from: 0x5B3...eddC4 to:  value: 0 wei data: 0x608...000ea logs: 0 hash: 0xe85...5ae69
status              true Transaction mined and execution succeed
transaction hash    0xe85038d9838c60a3344a65688ed244791771 6efdaed068b0ad99e05ad705ae69
from                0x5B380a6a701c568545dCfc803Fc8875f56bedd04
gas                 80000000 gas
transaction cost    168104 gas
execution cost      168104 gas
hash                0xe85038d9838c60a3344a65688ed244791771 6efdaed068b0ad99e05ad705ae69
input               0x608...000ea
decoded output      -
val                 0 wei
```

2. I have reduced the number of transactions in the "giveRightToVote" function while maintaining the same functionality of the program to save the gas, by doing the following:

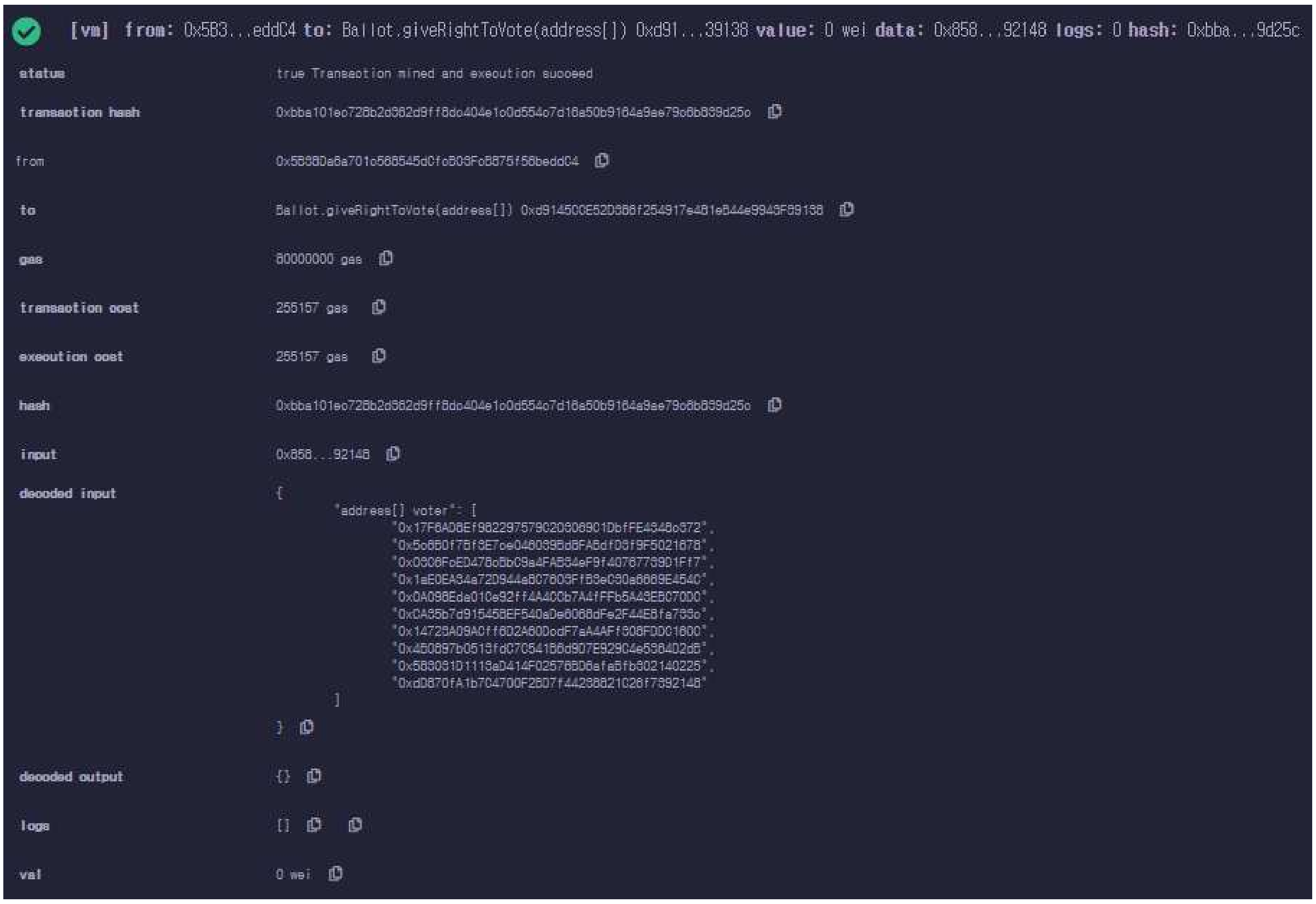
```
* Setting all voters voteCount to 1,
which means that initializing to (voteCount) be zero is not required.

* If the voter has already been casted the ballot,
one would be denied to vote at vote function.

* address array has been came, not playing with single address. batch job.
```

- When calling the `giveRightToVote(address)` in the previous solidity file, the call cost is 48857 each which means that running for 10 voters each would require 488570 gas around.



```
[vm] from: 0x5B3...eddC4 to: Ballot.giveRightToVote(address) 0xd91...39138 value: 0 wei data: 0x9e7...8c372 logs: 0 hash: 0x796...ec5f5
status              true Transaction mined and execution succeed
transaction hash    0x79676cc3cff54080a22150390a2a67eb57dd47b0e2edcbe70d27c036fddec5f5
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Ballot.giveRightToVote(address) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                 80000000 gas
transaction cost    48657 gas
execution cost      48657 gas
hash                0x79676cc3cff54080a22150390a2a67eb57dd47b0e2edcbe70d27c036fddec5f5
input               0x9e7...8c372
decoded input       {
                            "address voter": "0x17F6AD8Ef982297579C203069C1DbfFE4348c372"
                    }
decoded output      {}
logs                []
val                 0 wei
```

- However, when calling the `giveRightToVote(address)` function in Ballot.sol, it costs 255157 gas which saves 233413 gas.



```
[vm] from: 0x5B3...eddC4 to: Ballot.giveRightToVote(address[]) 0xd91...39138 value: 0 wei data: 0x858...92148 logs: 0 hash: 0xbba...9d25c
status              true Transaction mined and execution succeed
transaction hash    0xbba101ec728b2d362d9ff8dc404e1c0d554c7d18a50b9184a9ae79c6b839d25c
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Ballot.giveRightToVote(address[]) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                 80000000 gas
transaction cost    255157 gas
execution cost      255157 gas
hash                0xbba101ec728b2d362d9ff8dc404e1c0d554c7d18a50b9184a9ae79c6b839d25c
input               0x858...92148
decoded input       {
                            "address[] voter": [
                                    "0x17F6AD8Ef982297579C203069C1DbfFE4348c372",
                                    "0x5c6B0f7Bf3E7ce046039Bd8FABdfD3f9F5021678",
                                    "0x03C6FcED478cBbC9a4FAB34eF9f40767739D1Ff7",
                                    "0x1aE0EA34a72D944a8C7603FfB3eC30a6669E454C",
                                    "0x0A098Eda01Ce92ff4A4CCb7A4fFFb5A43EBC70DC",
                                    "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c",
                                    "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C",
                                    "0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB",
                                    "0x583031D1113aD414F02576BD6afaBfb302140225",
                                    "0xdD870fA1b7C4700F2BD7f44238821C26f7392148"
                            ]
                    }
decoded output      {}
logs                []
val                 0 wei
```