SMART CONTRACTS IN

SOLIDITY

PROGRAM

- What is Ethereum?
  - EVM
- What is Solidity?
  - Development Environment
- First Contract
- Assignments
  - Easy CR~~UD~~
  - Voting Machine

WHAT IS ETHEREUM?

- Decentralised single state machine
- Executes smart contracts
- Virtual Machine applies changes to state
- Ethereum’s purpose is not primarily to be a digital currency payment network.
- Ether, Gas, GasPrice

TRANSACTIONS

- A transaction is either completely executed, or not executed at all
- A transaction is a unit of work treated in a consistent and reliable way, independent of other transactions
- Only transactions can trigger a change of state or cause a contract to execute in the EVM

HASHING

- A hashing function is a function that gets an input and generates a unique output (but will be the same for the same input).
- From the output, the input cannot be calculated

- “Hello” –> 8b1a9953c4611296a827abf8c47804d7
- “hello” –> 5d41402abc4b2a76b9719d911017c592

BLOCKS

- A block is a combination of several transactions.
- A hash is calculated of the block and stored at the end of the chain
- Each block contains the hash of the previous block. This makes it impossible to change previous blocks without changing all subsequent blocks

ETHEREUM VIRTUAL MACHINE (EVM)

- Similar to other VM's (Java Virtual Machine, …)
- Handles smart contract deployment and execution
- Computes state transitions from smart contract executions
- Process of solidity smart contract to execution
    1. Write Solidity smart contract
    2. Compile to EVM bytecode (instructions that the EVM can read)
    3. Deploy EVM bytecode as contract to Ethereum network
    4. Execute EVM bytecode inside EVM

GETTING TO WORK

- https://github.com/karimStekelenburg/WorkshopSolidity

EXERCISE 1: SIMPLE BANK

- Create a contract with three functions: deposit, withdraw, and getBalance
- Make sure the deposit function can receive ether.
- Make sure the withdraw and getBalance function can only be called by the contract's creator.

- Extra: add a function that sends all the funds in the contract to an address and make sure it can only be called by the contract's creator.

EXERCISE 2: SIMPLE DATA STORAGE

- Create a contract with two functions: setPersonalData() and getPersonalData()
- Make sure setPersonalData() stores three variables: myName, myAge and myAddress into a PersonalInfo struct.
- Make sure the getPersonalData() returns the information stored in the PersonalInfo struct.

PROOF OF WORK

- Used to validate transactions and create new blocks
- Very difficult challenge for miners to solve, but easy for other miners to verify
- Low probability of solving the challenge makes it unpredictable which miner will generate the next block
- 51% attack
- So much energy

PROOF OF STAKE

- New way to validate transactions and create new blocks
- No challenges to be solved, save a lot of energy
- Miners will deposit ether into a smart contract. If a miner is deemed to be malicious the network simply locks this ether away
- 51% attack will be super expensive (because you need more than 50% of all Ether)