Blockchain Bootcamp Day 2

* 1st problem in eth app. Onboarding users.
* IN the past somehow replicate blockchain in end user computer, download all blocks in computer. 30gb of blocks
* Metamask to onboard new users.
* Private keys give access to all funds found in account.
* Main Ethereum network is where real network or Ethereum cost money.
* Then they are test networks that are separate network where Ethereum doesn’t cost money.
* Get test ether for testing applications.
* Sometimes faucets are not working on test networks
* Faucet Is a website that gives free ether to test.
* Follow transaction as soon as it’s mined.
* No difference between test networks from a developer perspective.
* Each network is difference the way the faucet deposit ether’s for each test network.
* After doing a transaction using a test network’s faucet, it will have a record of the transaction in the ethernet.

What Behind the Scenes

* Magic for metamask to interact with blockchain.
* Metamask uses infura(blockchain as service provider.) -> hosting Ethereum node.
* Using a restful api to interact with blockchain node.
* The browser used to get ether, communicates with backend with in turn communicates with blockchain node.
* There is a shared ledger between metamast and the restful api responsible for getting ether.
* EtherScan has a local copy of the blockchain and display results on browser window.
* Single source of truth.
* Other nodes can download block locally.
* Can use infura to interact with blockchain using a hosted node.
* Blockchain is the single source of truth
* Infrastructure providers that abstract running blockchain away from developers and users making it easy to use.

Cryptograph and Transactions.

* To send test transactions or transfer between account using meta mask
* Create another account.
* Transaction Fee:

How Ethereum transactions work

* Each Ethereum transaction object takes certain properties. (from: (account), to:(account), value: (integer in wei)
* Transaction fee that is regulated by gas price
* Data: ABI Byte string(optional)
* Nonce: a integar used everytime a transaction is sent.
* How does blockchain know a transaction isnot a malicious actor.
* How does it know to send funds from one account to another account
* It uses signatures, they built up. They sign the transaction object using web3.ethe.signTransaction. Takes a account that needs to be unlocked, then it can create a transaction signature.
* How does a transaction signature look like?
* It has new object with 3 new fields that are v, r, s.
* How does this work?
* Classical transaction object.
* Transaction has private key. -> 32 byte character long, 64 hexcharacter
* Can create private keys yourself in notepad. Must be random as possible with safe randomizer.
* Every private is different.
* Uses the ecdsa algorithm – can create a public key from private key, but can’t create private key from public key.
* This make thee blockchain secure.
* Take the public key and create Ethereum account. and create a Keecak hash of the last 20 bytes (last 40 hex characters) of the public key.
* Can go to private key-> public key-> eth account.
* Can’t go to account -> public key -> private key
* Not bad to go from eth account -> to public key but not to privatekey
* The private key and transaction are linked via a signed transaction.
* You would use the ecrecover function from signed transaction will output public key and Ethereum account.
* With signature can recreate public key. Therefore can retrieve underlying private key.(But can’t retrieve it.) The private key is the single source the user possess to sign transaction.
* Private key NEEDS TO BE KEPT SAFE>
* When private key is lost, can be used to transfer funds from one account to another account.

Cryptographic Hashing Introduction

* A typical hash function input string -> output digest.
* If one is changed in string -> totally different output digest.
* Ideal cryptographic hash function 5 properties

`.Is deterministic so the same message always results in the same hash. Different output based on input.

Very quick to compute hash for any message

It is infeasible to get input from output without brute force.

A small change in input changes hash value completely

Hash values should not collide. No two different hash value should be the same.

* You get all info from block and hash the information. Retrieve all transaction between this block and another rblock and hash them again. You know previous of the original block.
* Then another transaction, the transaction has hashed information of the sending block. This continues endlessly in a blockchain.They are copies of the same information of the nodes in the network. That is why it is so secure. All the copy’s of the blockchain needs to be changed.
* Hashing algorithm is ht emathematical foundation of blockchains.
* Blocks have hashed of the previous blocks’
* Are chained together.
* Changing information in prevous block changes all blocks thereafter.