**Vid 35 – Decentralization**

**At this point in our code 🡪 transactions originate, and blocks are added at a single point.**

A bank can certainly use a blockchain to store its transactions. The bank would keep a blockchain ledger and users will be free to ask for their balances and to add new transactions.

**Graphical user interface

Description automatically generated**

A picture containing text, clock

Description automatically generatedA good decentralized crypto is one for which every user can have their own copy of the blockchain. They can also publish transactions signed with their public key and assemble transactions into a block and add them to the block.

Our ecosystem will look like this. To begin were going to assume there are many users all w/ their own copies of the blockchain. Meaning that any user can tell how many coins are controlled by any public key at any time.

A screenshot of a computer

Description automatically generated with low confidence

When a user wishes to spend coins they sign a transaction and publishes it (broadcasts it to the rest of the network)

Also want to decentralize the adding of the new blocks to the block chain. We want anyone to be able to take the transactions she’s heard about, assemble them to a block and add them to the block chain.

Graphical user interface

Description automatically generated

**Vid 36 – the dark side of decentralization**

Due to the decentralization of the blockchain potential problems arise, such as the realities of moving info through a network, and nefarious actor trying to steal coins or other ways to cheat.

With many miners adding blocks to the network, we need a way to resolve the case in which two blocks have the same parent. If two blocks are added at or near the same time, there may be some confusion as to which block comes next in the block chain.

So, we can have a branch in what would otherwise be a linear blockchain. This branching is also a thing that hackers can take advantage of in attempt to steal coins

A picture containing diagram

Description automatically generatedImagine Alice sends coins to Henry and that transaction gets added to the block chain. Someone looking at this last block in the block chain will know that the coins now belong to Henry. But if Alice can prevent ppl from seeing that block, she can try to spend those coins again. For example, if she were to mine another block with the same parent as the block that involved her spend but leaving out her spend, she can keep control of the coins for anyone who sees her new block.

Here she’s trying to replace the block that says 10 coins go from Alice to Henry with a Block that says “10 coins go from Alice to Henry” with a block that doesn’t include that 🡪 later she is going to send those same coins to Clara.

**Double Spend Attack** 🡪 Clara receives goods from both Clara and Henry with the same coins.

**Vid 37 – Proof of Work**

Website

Description automatically generatedIf Alice is very persuasive, she might be able to add a branch to a very old block and erase a significant amount of history.

For this reason, we’ll write our code such that the minors only add blocks to the longest valid chain that they see. So, if Alice announces a new block that has a very old parent block, users might store that block, but miners will look for the longest chain and Alice’s block will be ignored.

A Branch in the block chain is not always a sign of an attack on our network. Sometimes two blocks are added very close to the same time. As new blocks are passed thru the network they may arrive in a different order at different nodes.

Diagram

Description automatically generatedIf most miners are added to the longest chain, branches are not impossible, but will likely be short and short lived.

To keep the blockchains code secure and make it trustworthy, it must be open sourced, so anyone can see the code and see if it abides by all regulations. Also means that anyone can modify it to disobey the requirement to add to the longest chain.

If bad actors were committed to eliminating a certain block, they could try to add blocks onto an alternative chain until that chain becomes the longest. This should be a problem unless a major portion of the blocks are added by attackers.

Cheap to set up many copies of modified nefarious software and overwhelm the network with billions of new blocks added to some branch chain.

One approach to avoid this is to have a network of nodes that you **trust** **and only add blocks from the nodes that you know are following the rules**. It is difficult to reliably establish trust in an anonymous and decentralized network

**The Byzantine Generals Problem 🡪** the problem of arriving at a consensus with the possibility of some attacking nodes.

**Graphical user interface, text, application

Description automatically generated**

**Proof-of-Work:** Under this scheme, to add a block to the block chain, the user is required to have done some computational work. Possible to set up many nodes but to add many blocks, requires a lot of computation.

**Characteristics of a good Proof-of-Work Algorithm:**

1. **Must monopolize a real-world resource for some amount of time**
   * Algorithms require many int or floating-point operations indicating that a user has monopolized the compute units of their computer
   * Other algorithms are memory heavy 🡪 setting up a problem that requires an amount of system memory to be occupied or a lot of data to be moved in and out of memory while doing a much lighter weight computation
2. **Proof must be easily verifiable** 
   * Node submitting new block should do a lot of work, but it should require very little work from all the other nodes to determine that the work has been done
   * b/c we don’t want to repeat all the work that the miner did to add the block
3. **The amount of resource required for each block needs to be variable over time**
   * For EX: at the beginning of bitcoin the work required for a new block could be done in a few hours on your home computer. Today, it takes several thousand nodes of the worlds fastest supercomputer to add a block in anything less than a day.
     + B/c of the large increase in the # of ppl mining
4. **The work needs to take the new block as its starting point**
   * If the work is not connected to a specific block, an attacker can just wait for a new block to be published, then copy the proof of work from that block and attach it to his own.
   * If the work isn’t connected at any specific block, how do you know which block is the real one?
   * **Pre-Computing the work 🡪** if the work were unconnected to the block, you can do a lot of pre-computation and have the proof of work done for thousands of blocks ahead of time, and then publish them all at once to overwhelm the network. NO POSSIBILITY OF PRE-COMPUTATION!!
     + Do this by making the block content the starting point for the computation

Why it helps to have a variable amount of work that you need to do to add each block?

A screenshot of a computer

Description automatically generated with medium confidence**Vid 38 – Nonces**

Hash takes a string of any arbitrary length and produces a hash of one specified length, this is what a hash of some data might look like.

A screenshot of a computer

Description automatically generated with medium confidenceThe Algorithm used by Bitcoin is called a cryptographic nonce

The proof-of-work algorithm used by bitcoin and others involves adding some bits to the end of a new block. They can be anything, but for the block to be valid, the hash of the block plus the bits must have a number of leading zeroes. These are called the “nonce” and this is the mining process.

As before the miner must verify that the transactions and the hash of the previous block are valid.

A screenshot of a computer

Description automatically generated with medium confidence**Miners** spend most resources on trying different nonces to see which one will give leading zeroes in the hash. Once the miner is sure that the transactions in the block are valid, he then tries a potential nonce and hashes the block + nonce. (Minimal work is done for transactions)

If the hash has the specified # of leading zeroes, he’s done.

**Chart

Description automatically generatedMore likely this potential nonce doesn’t produce enough leading zeroes**. So, the miner then adds a different nonce, hashes it, and looks for leading zeroes again. Miner keeps trying nonces until it sees a hash w/ enough leading zeroes to be valid.

Once found 🡪 it’ll publish that new block, including the nonce it’s found, to the block chain

A miner usually tries trillions of nonces before finding one w/ enough leading zeroes.

**Graphical user interface, text, application

Description automatically generatedComparing algorithm w/ list of requirements:**

* **Monopolizes a resource for a period of time (if hash is non-malleable)**
  + Non-malleable: 2 similar inputs hash to very dissimilar outputs.
    - If any changes made to input, we’ll have no info about the output.
    - Only thing can do is guess a new nonce, hash and check it
  + w/ a malleable hash might be possible to find a shortcut where the hash w/ 1 nonce gives a clue as to which to try next.
* **Easy to Verify**
  + Miner has to do trillions of hash computations looking for the right nonce but when a user is reading the block chain, he can verify the nonce w/ just **1 hash computation**
* **W/ nonce finding, we can vary the amount of work required by the changing # of leading zeroes that make a block valid.** 
  + Understand why necessary later
* **Block Content is a starting point for nonce finding**
  + Because were hashing the block + the nonce altogether not just the nonce itself.

**When mining in practice 🡪** important to constantly check for new transactions and blocks. If two miners are working to add a new block, they might have many of the same transactions in their new block. But what if one successfully adds a new block while the other is still working?

The remaining **miners must make 2 changes 🡪**

1. Update the previous block hash of their new block so that they are adding to the end of the chain.
2. When miners hear of a new block, need to remove any transactions that were included in the new block.
   * Including the same transactions in 2 different blocks results in an invalid block.

**Why is the variable “hardness” needed in algorithm?**

To convince these miners to do this work, buy the equipment, and spend all the energy they needed, there need to be an offer **REWARDED** each time they add a new block. Specifically, we allow miner to deposit a specified # of newly created coins to whatever address the successful miner chooses.

For our coins 🡪 allow our miners to add transactions w/ no input and output 25 coins.

* Miner will likely choose to deposit to own account

Letting miners create new coins, raises the risk of creating too many coins, and making # of coins in circulation unpredictable. (Like other currencies, the value of coins is reliant on there being a limited #)

To avoid this 🡪 crypto currencies aim to create coins at approximately a constant rate.

* Bitcoin produces 25 coins about every 10 mins. This is the emission rate
* it is kept relatively constant by adjusting the # of leading zeros required for a valid block.

**If coins are created too quickly 🡪** it usually means more miners are devoting more resources to mining. So, all the nodes in the network know to increase the required # of leading zeros in the hash and slow the emission rate again.

On the other hand, **if blocks are coming too slowly 🡪** the miners will decrease the # of leading zeros required to lift the emission rate back up.

* EX: if mining resources are being taken out of coin, maybe into another coin, maybe they’re doing some other kind of work

Table

Description automatically generated with low confidence**Quiz #4:**

Text

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

**Vid 39 – Assignment 1: Mining Rewards**

TxBlock 🡪 created a bunch or private and public key pairs, created some transactions, then checked those transactions for validity.

To allow miners to take rewards, gonna have to relax some of the requirements we have for valid transactions and for a valid block.

* EX: we specify that a tx always must have more or the same # of inputs that it has outputs. We have to relax that b/c we need to allow the miner to say, “this is my block, I get to add the reward to my own wallet (public key).”

Transactions coming in signed and gonna assume we’re a miner and we see these transactions coming over the network. We shouldn’t see miner using the private key at any point. He sees the tx, its already signed, he adds them to the block and then says I’m gonna add my own tx to collect my reward. Tx is gonna be flagged as invalid, according to our requirements, b/c it has more output than input

Graphical user interface, text, application

Description automatically generatedBlock reward fails, tx fee failed bc already invalid tx’s and the greedy miner was detected bc they were always invalid any block reward he would have taken would have been invalid, so the fact that its too high, we haven’t done it right yet

This is a check that as we try to fix and allow the miner to take these block rewards, we don’t want the greedy miner to slip through.

**Vid 40 – Assignment 1 solution**

For many coins when trying to make a tx it helps to have a higher tx fee that you’re offering to the miners b/c miners are looking for the most return on their investment.

Graphical user interface, text

Description automatically generated

for every tx we pull out the address and amount out of the inputs and add that to the total in

Text

Description automatically generated

**Vid 41 – Assignment 2**

Add the requirement to the blocks that the nonce produce leading zeros.

Going to look at how varying the # of leading zeros affects the time that is required to mine a new block.

**Graphical user interface, text, application

Description automatically generatedVid 42 – Python tips and tricks**

Discusses in-line lists and randint

Graphical user interface, text, application

Description automatically generatedText

Description automatically generated

Graphical user interface, text

Description automatically generated**Vid 43 – Solution 2**

Text

Description automatically generated

First were gonna make a list, in this list there is gonna be 1 zero hexadecimal character for of our leading zeros. So if leading\_zeros is 2 were gonna get 2 of these 🡪 \x00 and then were gonna join them w/out a separator

Text

Description automatically generated with low confidence

should give us 🡪