Foundations of Bitcoin

Shreemoy Mishra, IOV Labs

Pre-reqs for blockchain?

- None! 😆
- But technical background helps
- Touches a lot of areas
- Willingness to learn is all that's needed

Pre-reqs for blockchain

- Databases, cryptography, comm protocols
- software engineering, security, UI/UX
- economics: incentives, game theory
- social/legal: trust, securities, money transfer
- Hence: no strict pre-reqs

Questions?

- Remote presenations can be tricky
- If something goes off track ...
    - these slides (HackMd version)
    - https://hackmd.io/@optimalbrew/rJnko444v#
- https://github.com/optimalbrew/UCDavisBAF/
    - these slides (markdown)
    - simple example (ipython notebook)

This Meeting … I assume

- Very basic awareness of
- Cryptographic / Secure Hashing
  - **one way** functions (infeasible to invert)
  - “**unique**”" (no known collisions)
- Public Key Cryptography
  - Based on **private** and **public** keys

Secure Hashing Example

- Small changes lead to completely different output

```
>>> import hashlib
>>> m0 = hashlib.sha256(b'this is a simple example')
>>> m0.hexdigest() #the sha256 hash in hexadecimal
'b973545e5472e4a5b7570d65467b5ec3fd5a82195d1593a9815bb18ca425
>>> # add a character (a space)
>>> m1 = hashlib.sha256(b'this is a simple example ')
>>> m1.hexdigest()
'59a0b81d2d91800f86645ba3d0ee1f7a40322733edebfc9483b24e274a82
>>> # remove a character
>>> m2 = hashlib.sha256(b'this is a simple exampl')
>>> m2.hexdigest()
'4ca6aafe93b7ed4173421cb247119fbcfae66cca138d234f61ba3b861897
```

Secure Hashing Example

- Cryptographic (secure) Hash Functions

```
>>> m0 = hashlib.sha256(b'this is a simple example')
>>> m0.hexdigest() #the sha256 hash in hexadecimal
'b973545e5472e4a5b7570d65467b5ec3fd5a82195d1593a9815bb18ca425
>>>
```

- ❓ how long is a sha256 hash?

Public Key Crypto

- Based on **pair of keys**:
  -a *private key*, and a *public key*
- Uses:
  - authentication (e.g. SSH on Github, AWS)
  - **signing** (essential in bitcoin)
  - encryption

Public Key Crypto

- In Bitcoin: mostly concerned with **signing**
- Why not the other two?
    - authentication
        - Bitcoin is P2P, not client-server
    - encryption
        - **no secrets** in Bitcoin
        - All blockchain data public / not encrypted

Public Key Crypto

- Keys are very large integers
- `large (psuedo) random number -> private key`
- `private key -> public key`
- **crucial:** `public key` reveals nothing about `private key`

Public Key Crypto

- These large numbers are drawn from elaborate mathematical structures (*fields*, geometry)
- Special rules for (modular) arithmetic ($+$, $\times$), origin, *overflow* (e.g. 256 bit keys)

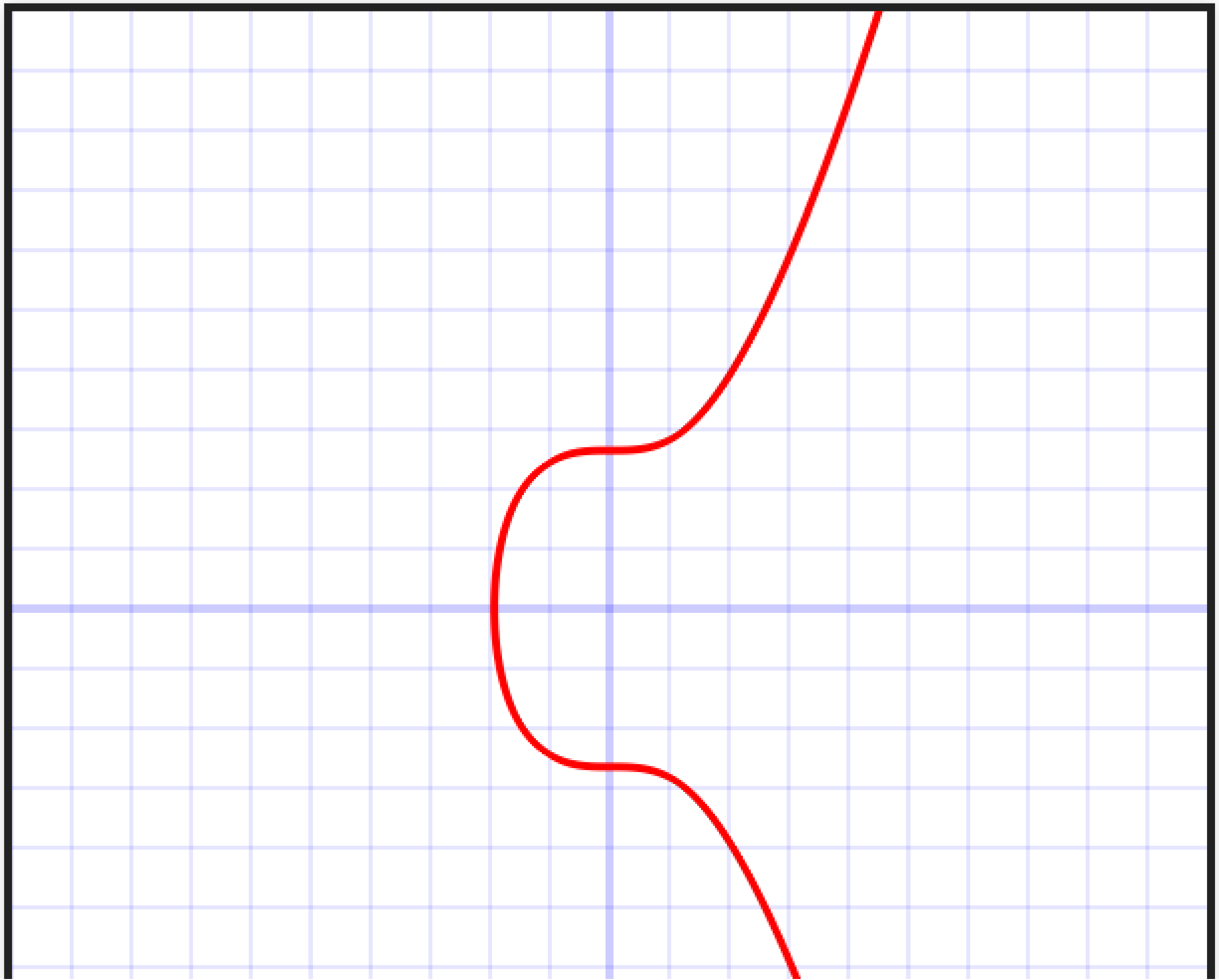
Digital Signatures Algo. (DSA)

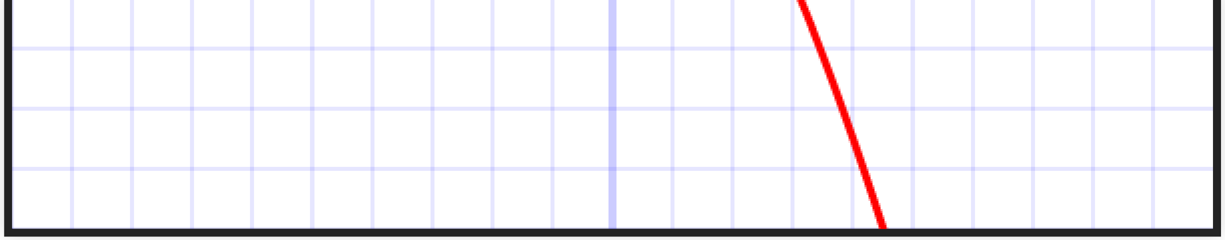
- Bitcoin uses Elliptic Curves (EC) cryptography
- ECDSA signatures consists of a **pair**:
- Typically denoted by r, s

Digital Signatures Algo. (DSA)

- Bitcoin uses Elliptic Curves (EC) cryptography
- ECDSA signatures consists of a **pair**:
- Typically denoted by r and s
- I was like what?? (the 1st time…)
- In ETH (and RSK), 1 more component v, so we have $< v, r, s >$

Public Key Crypto

- Using (simplistic) images of ECC can lead to flawed ideas
- 



Public Key Crypto

- We won’t discuss it
  - Leave it for cryptography
- Interesting aside:
  - in Bitcoin, each pvt_key -> 2 pub_keys!
  - hence 2 addresses
  - (what’s an address? coming up)

Signature Example (Go module)

```
package main

import (
        "crypto/ecdsa"          // key gen, sign, verify
        "crypto/elliptic"       // specific curve families
        "crypto/rand"           // PRNG
        "crypto/sha256"
        "fmt"
)

func main() {
    //generate key from secp256k1 curve for ECDSA
    privateKey, err := ecdsa.GenerateKey(elliptic.P256(), ra

    if err != nil { fmt.Println(err) }
```

Moving on to Bitcoin

Wiki https://en.bitcoin.it

- …first successful implementation of a **distributed crypto-currency** …
- …described in part in 1998 by **Wei Dai**…

Wiki https://en.bitcoin.it

- …first successful implementation of a **distributed crypto-currency** …
- …described in part in 1998 by **Wei Dai**…
- …using cryptography to control creation and transfer of money
- …**rather than** relying on **central authorities**. **Question:** what’s a central authority?

Bitcoin origins

- Shrouded in mystery
- Started by unknown individual(s) using a pseudonym: `Satoshi Nakamoto`
- Continuing speculation about identity, motives
  - not our concern today 😄

A P2P Monetary System (how fun!)

- Physical currency 💵 has *fixed denomination*
- E-money can have any value 10^{-8} BTC
- **Coins** are
  - chains of digitally signed messages
  - to transfer ownership
  - can have nearly arbitrary value
- Smallest unit in Bitcoin: 1 Satoshi

A P2P Monetary System

- You receive 2 coins: value 3.0, 4.0
    - you can create a coin worth 4.5 to pay someone
    - redirect 2.4 in a new coin to yourself
    - leftover (7-6.9=0.1) is transaction fee
- Will revisit this example in a bit

Fundamental security problem

- “**Double spending**”
- I pay you 100 coins (e.g. game/src/)
  - 1st signed message
- later, I transfer the *same* coins to myself
  - 2nd signed message
- I (try to) convince others that 1st msg **is invalid**

Fundamental security problem

- If not for this problem (double spending), e-currency is easy
- Goal:
  - Avoid multiple histories (conflict messages)
  - Agree upon a *single history* of payments

Basic jargon … Bitcoin Address

- Addresses are generated from public key

```
Key hash = Version + RIPEMD-160(SHA-256(public key))

Checksum = 1st 4 bytes of SHA-256(SHA-256(Key hash))

Bitcoin Address = Base58Encode(Key hash concatenated with Che
```

- **Not** double Sha256, checksum, and Base58

Basic jargon

- Transaction (shorthand TX)
    - a transfer of coins
    - from 1+ addresses (**TX inputs**)
    - to 1+ addresses (**TX outputs**)
- How much BTC do yo have?
- The net sum of all your *unspent* TX outputs

Basic jargon - Block

- A collection of transactions processed around the same time
- like a container class
- 1 block created approx every 10 mins
    - Who creates them?

Basic jargon - Block Chain

- **Chain** blocks together by including
    - a **pointer to previous (parent) block**
- similar to a *linked list* (single, not double)
- Done using a **secure hash** (of data in parent)

Basic jargon - Block Chain

- Chaining block hashes together…
- … makes it practically impossible to change info without being detected.
- important, because the goal is to agree upon a single historical record of transactions

Basic jargon - Block Chain

- Chaining block hashes together…
- … makes it practically impossible to change info without being detected.
- important, because the goal is to agree upon a single historical record of transactions

TX: recall example

- Receive 2 coins: value 3.0, 4.0
    - you can create a coin worth 4.5 to pay someone
    - redirect 2.4 in a new coin to yourself
- 2 TX inputs: (TXIs)
    - one for 3.0 and another for 4.0
- 2 TX outputs: (TXOs)
    - one for 4.25 and another 2.4

TX inputs and outputs

- $\sum TXOs \leq \sum TXIs$
- Any difference is collected by **miners** (block creator) as **fee**

Examples using block explorers

https://www.blockchain.com/explorer

- or any explorer
- search for block no 650750
- select BTC (bitcoin) chain (depending on explorer)
- https://www.blockchain.com/btc/block/650750

Checkout block info

on https://www.blockchain.com/btc/block/650750

- **Hash** = starts with lots of 0's
- Number of TXs = 2834, Size = 1,396,390 bytes
- **Timestamp** = 2020-09-30 20:11
- **Merkle Root** = d349...
- **Version** = 20, **Nonce** = 25,376,585
- Block reward = 6.25, Fee = 0.592 BTC
- Difficulty = 19,314,656,404,097.00, **Bits** = 386,831,018

Explorer

❓ What's average fee per TX? (in USD?)

❓ How would you find the hash of the **previous** block?

❓ what is the smallest block height/number?

Genesis: Block 0

- Find block 0 in explorer : https://www.blockchain.com/btc/block/0
- Look at Inputs of Coinbase TC
- Sigscript (some of you know what's coming)
  - paste the numbers starting from 5468 … into a Hex to Ascii converter
- https://www.rapidtables.com/convert/number/hex-to-ascii.html

Time to meet Satoshi
(just the paper)

“The” Bitcoin Paper

A Peer-to-Peer Electronic Cash System

https://bitcoin.org/en/bitcoin-paper

Abstract

- **Motivation**: Purely p2p e-cash payments
- Without banks or financial intermediaries
- **No need to trust** anyone

Abstract

- Digital signatures are necessary...
- but not sufficient ...
- because of the possibility of *cheating* (double spending problem)

Abstract

- **Solution:** transaction timestamps … so only the 1st one counts… rest are discarded
- ❓ how to reach **consensus** on timestamp when (block) data can arrive in different order?

Agreement on TX orders

- Timestamping TXs to agree on a system for **single history**
- Create **cryptographic links between timestamps**
- Make them impractical to falsify

Agreement on TX orders

- Timestamping TXs to agree on a system for **single history**
- Create **cryptographic links between timestamps**
- Make them impractical to falsify
- (Note:. impractical… not impossible)

Agreement on TX orders

- This record (linked hashes) cannot be changed (easily)
- The **difficulty of re-writing history** is what **blockchain security** is all about

Blockchain *can* be modified

- But to modify the record
- … one **needs to overcome** all the associated computational work that went into constructing the chain …
- … since the timepoint from which they wish to rewrite it
- Such changes are called **chain reorgs**

Wanna re-write history?

- Suppose you want to change a **single block** from Oct 2019.
- You have to redo the **entire chain**… from that point on.

Wanna re-write history?

- Currently it costs $10 million a day to create blocks (entire ecosystem) …
- You need to burn through at least 1/2 of that… and then some
- The math is not quite right… (I am skipping stuff)

re-writing history

- But even if you do that, all you can do is double spend some of your own coins.
- Is the cost worth it?

re-writing history

- But even if you do that, all you can do is double spend some of your own coins.
- Is the cost worth it?
- Not for BTC … but possible for other blockchains (ETClassic)

Majority is tied to longest chain length

- As long as **the majority** of computational power **is not attacking** the network…
- … the honest nodes will (probabilistically) outpace any attackers…
- … and generate the **longest chain** (or **“heaviest”**)
- Satoshi provides examples w/ C code

nodes come, nodes leave

- Nodes can go offline and rejoin at will…
- … and accept the longest PoW chain
- … as evidence of what happened while they were gone.

Transactions

- *Define a bitcoin* (or any coin) as - **a chain of digital signatures**
- Each owner transfers the coin by
    1. using their private key to sign (a hash) of the **previous transaction**
        - (where they got the coin from!)
    2. and **add public key** of next owner

Transactions

- To clarify… you don't have to do anything to receive coins
- Conditions are on spending

Transactions

- Anyone can verify the signatures and chain of ownership
- **Main problem: double spending**
    - recipient cannot verify that the signed coin has not been spent before…

Double spending

There should be no previous transaction

- with the same coin
- that someone is offering you now.

Double spending

- The only way to be sure …
- is to keep a record of ALL PAST transactions.
- which is why running a fully synced bitcoin node offes max protection
- but not practical for most people

Timestamp server

- Satoshi’s solution? **A p2p time server**
    1. Takes a block of TXs to be timestamped
    2. computes **a hash** of data from block
    3. widely distributes it across the network

Timestamp server

- This **block hash** helps proves that the transaction was known when timestamped
- Proves ❓ (coming up... Merkle Trees)

Timestamp server

- Each timestamp (hash) **also includes a hash of the previous timestamp (hash)**
- … re-inforcing ones before it… forming a chain…

Timestamp server

- BTW: the hash *is* the timestamp
- It is okay to use block number or height (unique) to talk about a TX's timestamp…
- But no one uses the date-time format

Questions before we move to some implementation details?

Coming up… proof of work

Implementing the system: POW

- Creating a block requires **guessing** a value (the **nonce**)
- which when combined with other data
- and **then hashed**…
- gives a number that **begins with a certain number** of 0's

Block Hash (double hash)

- PoW is a double-hash `sha256(sha256())`
- But what data are we hashing?
- **6 fields** from the block:
  - 3 integers: **version, bits, nonce**
  - 2 hashes: **previous block, TX Merkle root** (explain later)
  - The block's **time stamp** (UTC)
- Apart from nonce: limited degrees of freedom

Implementing the system: POW

- Nonce is 4 bytes: 2^{32} combinations, approx 4Billion
- Sha256? 2^{256} combinations
- The computational work is **exponential in the number of zeros**...

Implementing the system: POW

- finding a solution is very hard... lots of combinations
    - verifying a valid solution is trivial!
- It is like playing the lottery or mining for gold

Block Hash (double hash): format

- **Order**: version, prev_Hash, Merkle_Root, timestamp, bits, nonce
- Format
  - **hexstring** + **big endian**
  - concatenate (order)
- After double sha -> convert back to **little** endian
- Let’s go through an example

Implementing the system: POW

- PoW is how **“someone”** gets to create the next block
- and collect rewards…

Implementing the system: POW

- PoW is how **“someone”** gets to create the next block
- and collect rewards…
- but PoW is also about reaching consensus!

Implementing the system: POW

- PoW is also how **everyone** can reach agreement (**consensus**) …
- i.e. that a solution has been found…
- and move on to working on the next block
- … not everyone *has to* agree… but if a majority do, then that chain gets longer

Implementing the system: POW

- This resembles a majority voting system
  - where **1 hash computation equals 1 vote**
  - (in Satoshi’s words, one-CPU-one-vote)

Implementing the system: POW

- All decentralized blockchains have some voting mechanism…
  - next to PoW, the most well known is Proof of **Stake**…
  - where voting power is based on staking coins…
  - each coin staked is one vote (See Eth2.0)

Implementing the system: POW

- How do miners “vote” for a solution?
- by starting to work on the next block
- AND pointing to the accepted solution (block hash) - as the parent

Implementing the system: POW

- How many starting 0’s are enough?
- that depends on a parameter called **difficulty**
- difficulty is readjusted periodically
- using an algorithm that tracks a moving average of the last 2K blocks
- and keep that average around 10 minutes.

Implementing the system: POW

- In contrast… RSK block time is about 30 seconds.
- RSK is tied to BTC (security via **merged mining**)
  - something you may learn later
- RSK difficulty lower… only 1 in 20 RSK PoW meets Bitcoin's difficulty

Money creation! Block rewards

- When a miner creates a new block, they earn a new coin
  - 6.25 BTC (block reward)
  - TX fees (variable)
- the reward coin is implemented as a special transaction
  - called the **coinbase** transaction
  - By convention, the 1st TX (position 0) in the block

Money Supply

- Rate of new coin creation is called *inflation*
  - different from economics
  - Econ: inflation (π) is about price level
- In BTC, the number started with 50 BTC per block
  - reduced by 1/2 every 4 years (210K blocks)
- Using geometric series formula
  - theoretical max of 21 million BTC.

Block rewards ... miner incentives

- Recall the basic security problem...
- **if someone has majority compute power**
  - they can spend their own bitcoins more than once...
  - if this happens repeatedly... bitcoin loses value
  - miners will lose too

Block rewards … miner incentives

- However, if they have that much power,
  - they are better off just mining new blocks.
  - reward is an important for incentives for honest behavior.

Block rewards ... miner incentives

- and it is worth repeating...
- even if someone has all the majority hash power...
    - they cannot steal other people's coins ... why ❓

Stop for a sec for Q’s

(next stop… verifying your payment (TX) was included in a block)

- Given:
  - the TX’s hash
  - and the block number where it was included

Simplified Payment Verification

- How to verify that a TX was processed without running a full node.
- this is important, for example, for smart phone based apps and wallets

Simplified Payment Verification

- Two main ideas:
- store only the **block headers**
  - (80-100 bytes each)
- Use **Merkle** proofs

Merkle Trees

- Binary **hash** trees
- Leaves are TX hashes
    - TXs are identified by their hashes
- Each parent is the hash of it chldren

`sha256(sha256(L + R))`

- Odd number?? (no right child?)
    - repeat the last one
- Repeat until we get to root
    - **root included in Block header**

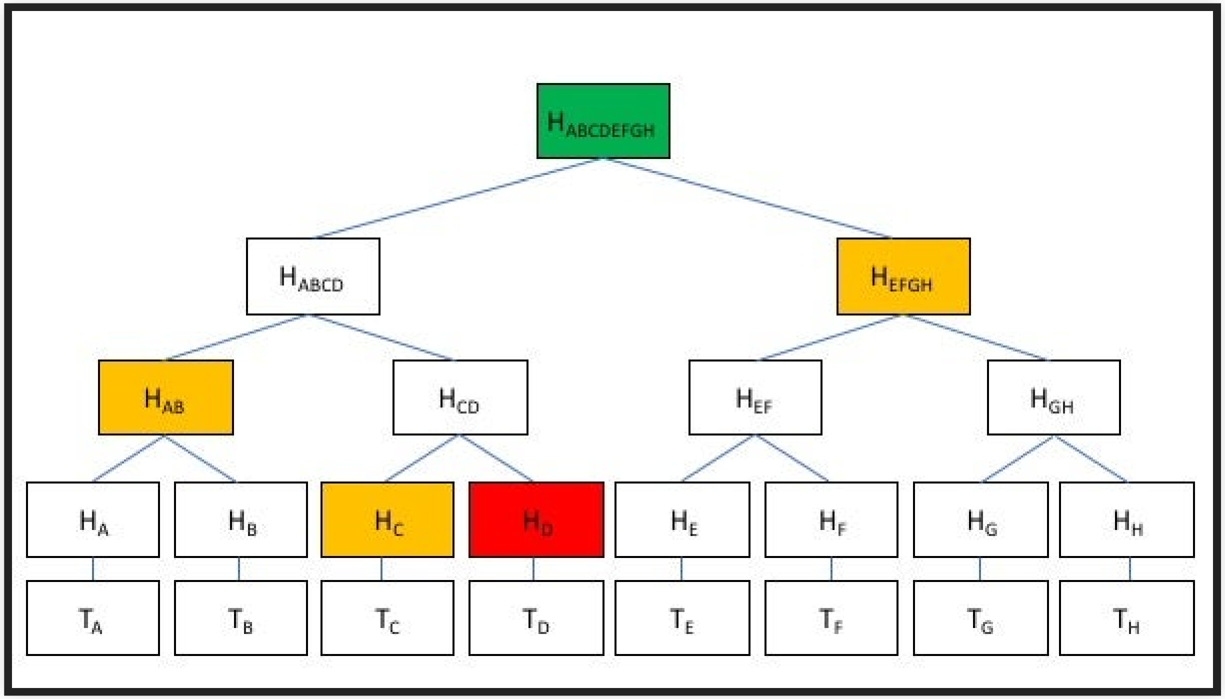
Merkle Trees

- Example: Coinbase TX + 2 user TXs:
  - Hashes: H0, H1, H2
  - Just 3 leaves in this binary tree (will have to repeat one)
  - Hashes: H0, H1, H2, H2 (again)

Merkle Trees

- parent of first two
  - `L = sha256(sha256(H0+H1))`
- parent of next ‘two’:
  - `R = sha256(sha256(H2+H2))`
- One level up? just the root in this example
  - `merkle_root = sha256(sha256(L+R))`

Merkle Tree and TX verifiation

- 

  - Image source: investopedia.com (… watch out… confusion… leaves are hashes!)

Merkle Tree and TX verifiation

- Watch out… potentially confusing…
- Leaves are just the hashes H_A, H_B …
- TXs: T_A, T_B … just for illustration…
- Otherwise, we don't have a binary tree, do we?
- Repeating: last layer with T_i's'… NOT part of tree
- Images from other sources may be different

SPV

- On top of the “Merkle proof” of TX inclusion,
    - we also wait a certain number of blocks to be added on top of the one where our TX.
    - often 6 blocks… less for small amounts
    - a lot more for serious amounts (100 for RSK-BTC bridge!)

Merkle Trees

- ❓ What if a block has no TX? e.g. block 0, 1?
- Someone mined a block with no TXs! (Can happen)
- What’s the Merkle root in that case? How big is the tree?

Merkle Trees

- Merkle trees are used for other things too…
    - distributed file storage (IPFS, Swarm, libtorrent)

Proof of Work illustration

Network: Tie breaking

- A node in Japan may receive a block from China faster than a block from Ireland
- more than 1 valid block … which one to vote for?
- ❓ what do you think happens?
- the program starts working on the next block, and points to whichever valid solution it received first.
- But it **does not** throw away the other one!

Network: Tie breaking

- Instead, it starts a **fork** (in the chain)
- as more blocks come in… some will point to one and some to the other…
- over time… one of the forks will grow “longer” (based on majority)
- when a node switches to a new fork (because it has more cumulative PoW), that’s called a **reorganization** or *reorg*

Network: Tie breaking

- such short-lived hard forks happen all the time
- Which is why, depending on the sums involved, the receipient of a transaction should wait a certain number of blocks before treating it as final.
- A common rule of thumb is 6 blocks… or about 55 minutes
- For small sums, you may choose to wait 10-15 minutes.
- If you are paying … it does not matter :)

Bitcoin mining involves

- **Reapeatedly compute hashes** of …
- a block header template + nonce
- until resulting hash is *smaller* than a **target**
- **before** someone else does (a race)

How is target selected?

- Probability a Hash satisfies the target

$$\frac{1}{D \times 2^{32}}$$

- D is difficulty (was 1 when it started)
- why 32? 32 comes from 32 bytes, i.e. 256 bits
- Readjusted periodically (2 weeks)

Payoff from mining

With simple assumptions

- constant hashrate h
- mining for time t
- constant difficulty D
- will find $ht/(D * 2^{32})$ blocks

Payoff from mining

- earn reward B per block
  $htB/(D * 2^{32})$
- neglecting fees

Example

- Buy dedicated mining computer
- capable of 1GH per second i.e.
- Mine for 1 day = 86400 seconds
- Difficulty $D = 169200$
- Block reward $B = 6.25 BTC$
- Daily earnings

$$\frac{10^9 * 86400 * 6.25}{169200 * 2^{32}}$$

Pooled mining

- Very high variance in solo mining
- Income smoothing via pooled mining
- Shares: used for accounting
- Shares are hashes that meet a lower target (hence more frequent)

Professional Mining

- Bitcoin mining
  - started CPU, then GPU, then FPGA + ASICs
- https://bitmain.com/product/antminer-s19-pro-110th-s/ 3250W

Professional Mining

- ASIC: https://bitmain.com/product/antminer-s19-pro-110th-s/ 3250W
- like 2-3 space heaters on full power
- weighs about 35 lbs
- $ 2500 purchase +
- electricity cost: about $10/day in Seattle
- 110TH (Terahash)
- BTC network about 150EH (ExaHash) [Giga->Tera->Peta->Exa]

Mining Costs (energy)

- So one S19 pro is less than $1/10^6$ network hashing power
- A probability of less than 1 in a million
- Alternate view: $10M a day in elec costs
- 24 hours * 6 blocks = 144 blocks * 6.25 BTC * $11K = $9.9M
- so our energy estimate is somewhat higher, but same order of magnitude.
- Also: neglected TX fees in payoff
- Also: Hashrate is not constant… it is a market … miners come online and go offline based price of BTC and electricity.

Spending scripts

- In explorers, you will often see scripts for spending UTXOs
- `OP_HASH160 OP_DUP OP_EQUALVERIFY <PUBKEYHASH> OP_CHECKSIG`

Spending coins P2PKH

- Pay to public key hash
- when redeeming / spending a UTXO…
    - provide PubKey
    - also provide signature (which will be matched with pubkey)
    - don’t use same pubkey again (wallets do this automatically)
- PKH is shorter than PK (orig script was P2PK)

P2SH

- Pay to Script Hash
- The recipient provides the redeem script
- Address starts with 3 (instead of 1)
- As sender, you know nothing about how the address is controlled
- How to spend?
  - provide a script whose hash matches the `ScriptHash`
  - provide data (for computing) so the script evaluates to `True`
  - look for examples online

Information sources

- Satoshi's original: https://bitcoin.org/en/bitcoin-paper
- Community site: https://bitcoin.org/
- Bitcoin Wiki: https://en.bitcoin.it/wiki/Main_Page
    - Largely free of bias (crypto projects can be very opinionated)
    - outdated links and code samples
- code: https://github.com/bitcoin/bitcoin
- Mining (pools): Meni Rosenfeld paper
    - Why? part of the public conversation
- Both papers: easy and fun to read