Table of contents

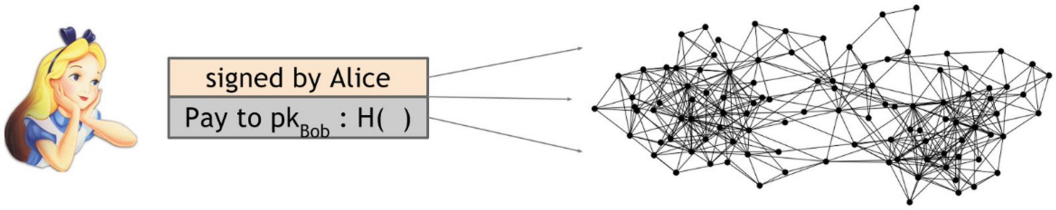
- Distributed consensus
- Consensus without identity using a blockchain
- Incentives
- Hash puzzles
- Bootstrapping

Distributed consensus

- Key technical problem in building a distributed e-coin system is achieving distributed consensus
  - Decentralizing ScroogeCoin
- Distributed consensus has various applications
  - Has been studied for decades
- *n* nodes | each have an input value | some faulty nodes
- A distributed consensus protocol has following properties:
  - It terminate with all honest nodes in agreement on a value
  - The value must have been generated by an honest node

Distributed consensus

- In the context of Bitcoin
  - To pay Bob, Alice broadcasts Tx to the entire Bitcoin



  - Bob's computer is nowhere in this picture
  - Other are broadcasting too
    - Nodes must agree exactly on
      - Which Tx were broadcast
      - Order in which these Tx happened
    - If nodes agree, then we have a single, global ledger

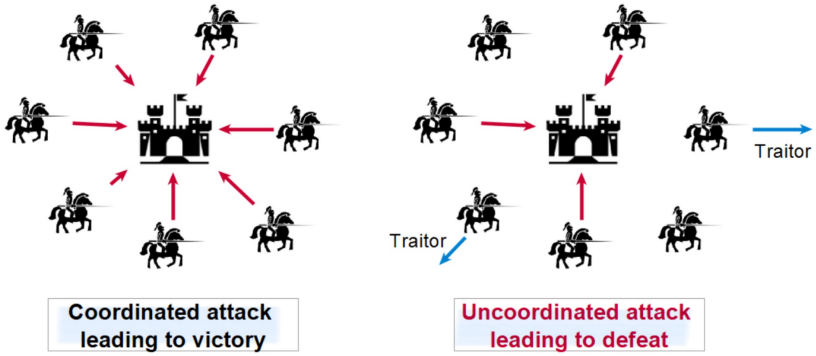
Nodes should agree on the transaction broadcasted and order in which it took place, I.e. Alice to Bob, then Bob to Rambo and so on. If agreed then we have a single ledger or blockchain on which the entire system agrees.

Distributed consensus

- In the context of Bitcoin
  - When to come to consensus on a block?
    - At regular intervals, say every ~10 minutes
  - There's more to complicate the scenario
    - Imperfections in the network (latency and crashing)
    - Deliberate attempts by malicious nodes
  - No notion of global time
    - Can't do ordering based on timestamps
      - Remember "*Timekeeping is not of much importance in Bitcoin*"

Distributed consensus

- Byzantine Generals Problem



Coordinated attack leading to victory

Uncoordinated attack leading to defeat

Distributed consensus refers to the process by which multiple, independent participants in a network agree on a single, consistent state or decision, even if some participants may fail or act maliciously. In the context of systems like Bitcoin, it ensures that all participants agree on the valid set of transactions (the blockchain) without relying on a central authority.

Bitcoin deals with the "Byzantine Generals Problem"—a problem in distributed systems where participants must agree on a common decision despite the possibility of some participants acting maliciously or failing to communicate properly.

- How Bitcoin deals with it?
  - Introduces incentives
    - Novel for a distributed consensus protocol (it's a currency)
  - Doesn't solve distributed consensus problem entirely
    - But, solves it in context of a currency system

Consensus without identity using a blockchain

- If nodes had identities, the design would be easier
  - *The node with the highest numerical ID should take some step*
- No central authority to assign identities

But this will make the system centralised indirectly. And in first place there is no concept of nodes having identities in bitcoin.

  - Sybil attack
    - Adversary creates sybils/copies of nodes
      - Controlled by adversary
    - Influence the system by number
- Prevent adversary from multiply his power by creating new nodes

Consensus without identity using a blockchain

- Bitcoin consensus algorithm (simplified)
  - New transactions are broadcast to all nodes
  - Each node collects new transactions into a block
  - In each round a random* node gets to broadcast its block
  - Other nodes accept the block only if all transactions in it are valid (unspent, valid signatures)
  - Nodes express their acceptance of the block by including its hash in the next block they create

*not vulnerable to Sybil attacks **Due to random selection of nodes that broadcast the blocks.**

Consensus without identity using a blockchain

- Stealing Bitcoin: No!
  - Create a valid transaction
  - Forge the owners' signatures

To create a valid transaction we require owner's private key, and forging someone's private key using his public key (which is available publicly) is infeasible using the current technology.

- Denial-of-Service (DoS) to prevent Tx from a user: No!
  - Some honest node will pick the Tx

this kind of attack is difficult because there are many honest nodes in the network. Even if some malicious nodes ignore the transaction, some honest node will likely pick it up, validate it, and propagate it through the network.

- Double spend attack: Depends!

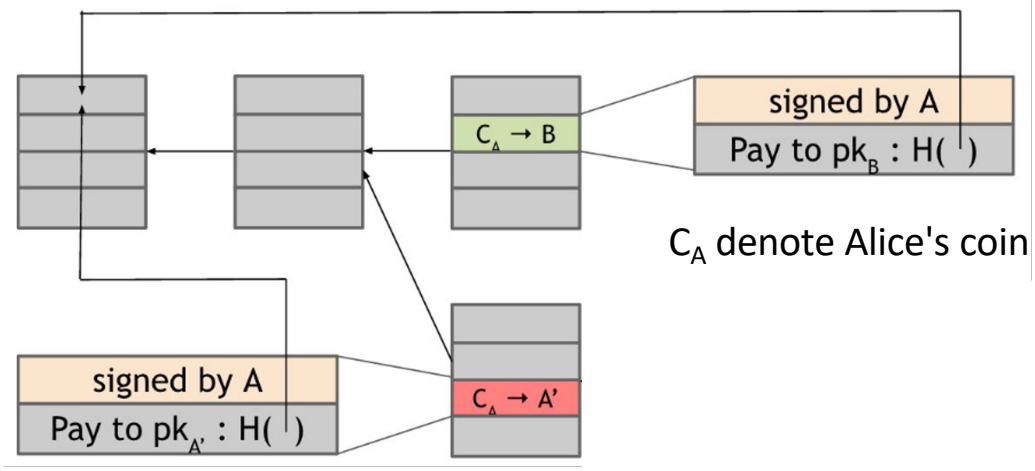
- Two different types of hash pointers in Bitcoin
  - Blocks include a hash pointer to prev block that they're extending
  - Tx include >=1 hash pointers to prev Tx outputs that are being redeemed

if Alice is sending bitcoins to Bob, her transaction will reference (via a hash pointer) the outputs of previous transactions where she received those bitcoins. This allows the network to verify that Alice actually owns the bitcoins she is trying to spend.

Consensus without identity using a blockchain

- Double spend attack
  - Alice creates two Tx
    - One to Bob
    - Another to herself (address), double spends

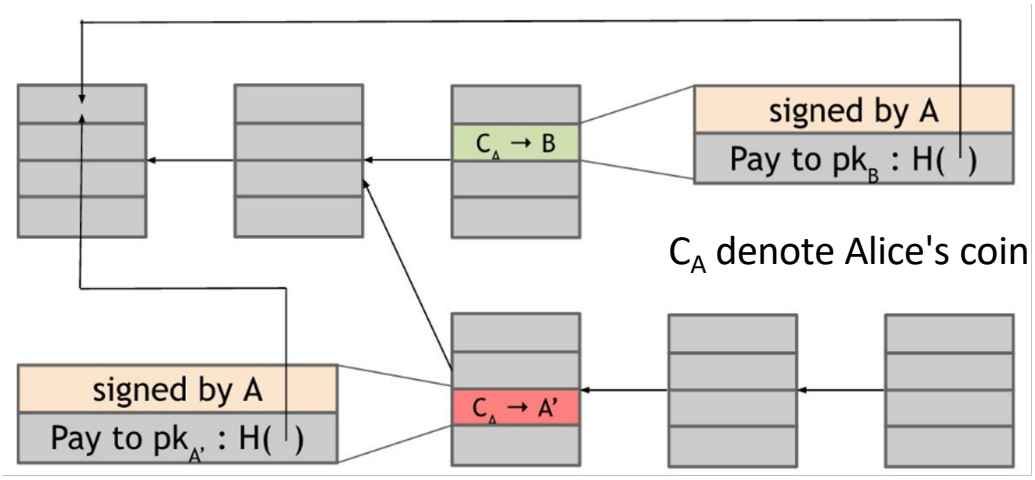
Notice carefully, Alice has created two transactions for double spending, they both are picked by two different blocks , which are pointing to the hash of the same previously accepted block. And the transaction hash they are pointing to are also the same somewhere in the past, from where the bitcoins are redeemed(nothing but the second hash explained above).



  - Only one Tx can be included in the blockchain

Consensus without identity using a blockchain

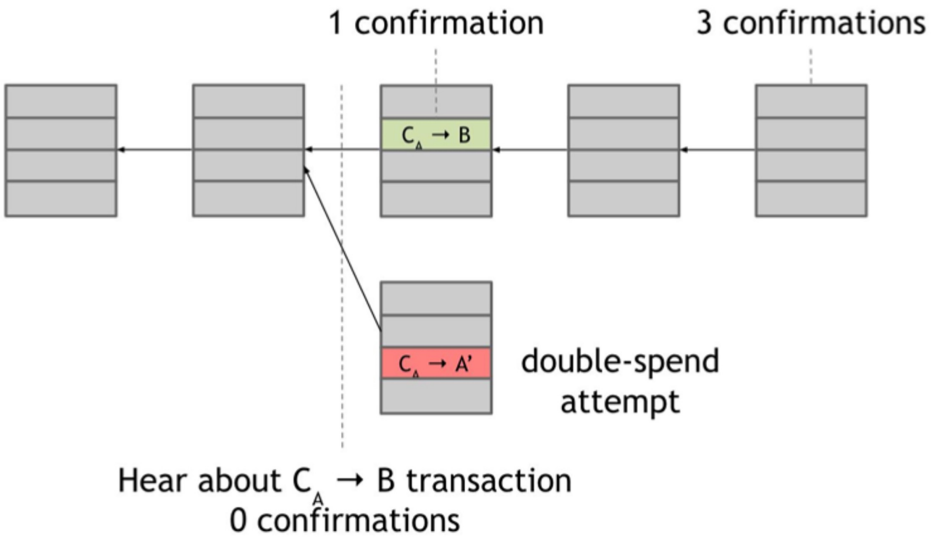
- Double spend attack
  - Will double spend attempt be successful?
    - Depends on which block will end up on long-term consensus chain



  - Network totally ignores block with Bob's Tx, an ~~orphan~~ stale block

Consensus without identity using a blockchain

- Double spend attack
  - Zero-confirmation transaction



Consensus without identity using a blockchain

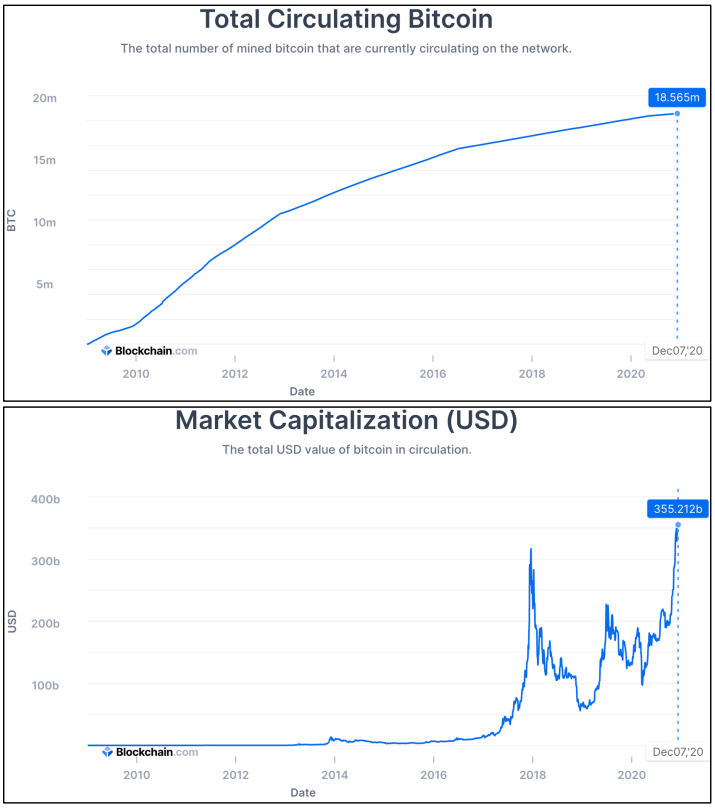
Cryptography means signing the transaction with private key. Consensus means miners validating the block.

- Recap
  - Protection against invalid Tx is entirely cryptographic
    - Enforced by consensus
      - Honest nodes prevent including a cryptographically invalid Tx
        - Honest majority assumption
  - Protection against double spending is purely by consensus
    - Cryptography has nothing to do here
      - 2 Tx of double spending attempt are both cryptographically valid
    - Consensus decides which ends up on long-term consensus chain
  - You're never 100% sure that your Tx is on the consensus branch
  - After ~6 confirmation, virtually nothing can go wrong

Incentives

- Block Reward
  - Reward for successfully creating a new block
    - Initially, block reward was 50 BTC/block
    - Reward halves every 210,000 blocks (~4 years)
    - Current reward (Dec '20) is 6.25 BTC/block
    - Only way to create new Bitcoin
      - Controlled supply
      - Pre-defined in Bitcoin protocol: Max 21 million BTC
- BTC is Bitcoin ticker symbol
- Smallest value = 10^{-8} BTC = 0.00000001 bitcoins = 1 Satoshi

Incentives



Incentives

- Block Reward
  - New block creation reward will run out in year ~2140
    - The system will stop working in 2140?

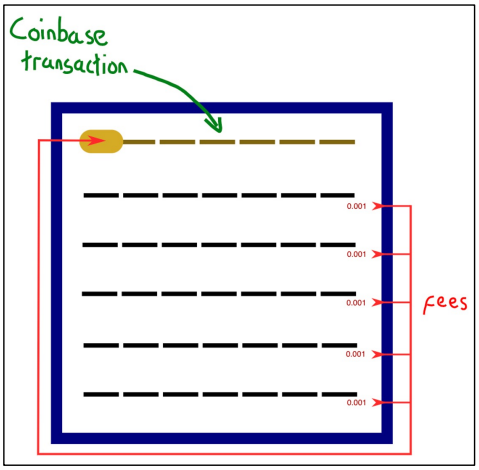
Incentives

- Transaction fees
  - Second incentive mechanism
  - Tx sender may pay a small fee
    - Tx fee is purely voluntary
  - Sender can choose to make $Value_{Out} < Value_{In}$
    - Block creator that first puts this Tx into blockchain gets the difference
      - Tx fee = sum(inputs) - sum(outputs)

Value out < Value in : It's like sender is sending 1000100 satoshi and he is actually intended to send only 1000000 satoshi to the receiver. Thus 100 satoshi is the transaction fees which the miner which puts that transaction on blockchain gets.

Incentives

- How to collect block reward and Tx fee?
  - A coinbase Tx
    - First Tx in a block
    - Unique type of Tx created by block creator
    - Used to collect both block reward + Tx fees



Incentives

- How to collect block and Tx fee reward?
  - A coinbase Tx
    - Single blank input (zero input value + no reference to prev Tx)
    - One per each new block
    - Before or after block is accepted?

Inputs

Index	0
Details	Output
Address	
Value	N/A
Pkscript	N/A
Sigscript	Invalid ASM
Witness	0000000000000000000000000000000000000000000000000000000000000000

version	02000000
previous block hash (reversed)	17975b97c18ed1f7e255adf297599b55330edab87803c817010000000000000000
Merkle root (reversed)	8a97295a2747b4f1a0b3948df3990344c0e19fa6b2b92b3a19c8e6badc141787
timestamp	358b0553
bits	535f0119
nonce	48750833
transaction count	63
coinbase transaction	
transaction	
...	

Incentives

- How to collect block and Tx fee reward?
  - A coinbase Tx
    - ~~Orphan~~ Stale blocks?
      - Bitcoin from a coinbase Tx are spendable after 100 confirmations

Hash puzzles

- Two questions still remain
  - Who will create the block?
  - Adversary might create a large number of Sybil nodes
    - To subvert the consensus process
    - Get incentives, a free-for-all reward capturing race

- Problems are related
  - Same solution, called Proof-of-Work (PoW)
    - Select nodes in proportion to a resource that nobody can monopolize
  - Resource = computing power > PoW system, Bitcoin
  - Resource = ownership of the currency > Proof-of-Stake (PoS)

Means select nodes in proportion to the resources for computing that is not possible for the attacker to overpower and monopolise the system.

PoW (Bitcoin): Nodes use computing power to win the right to create blocks.
PoS: Nodes are selected to create blocks based on how much currency they own.

Hash puzzles

- Bitcoin achieves PoW using hash puzzles
  - To create block, node must find a random number (nonce), such that

$$H(nonce \parallel prev\_hash \parallel Tx \parallel Tx \parallel \ldots \parallel Tx) \leq target$$

  - If hash function is puzzle-friendly
    - Only way to solve hash puzzle is to try enough nonces one by one
    - Repeatedly trying & solving hash puzzle is **Bitcoin mining**
    - Such nodes are **miners**
      - Mining cost = H/w cost + operation cost (electricity, cooling, etc.)

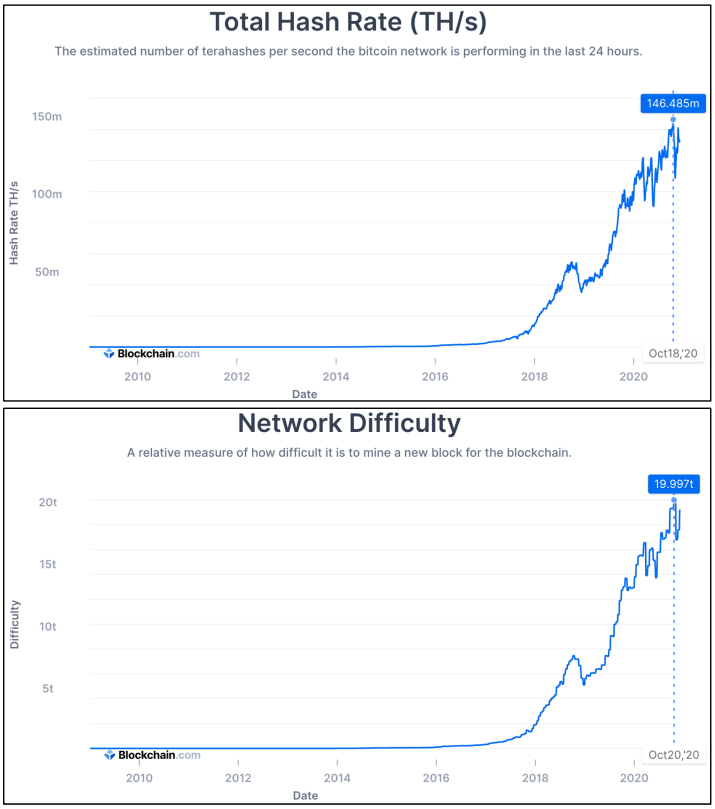
Hash puzzles

- Hash puzzles properties
  - Difficult to compute
    - Anybody can be a miner
      - Puzzle shouldn't be easy enough
  - Parameterizable cost
    - Shouldn't be a fixed cost for all time (computing power is increasing)
      - Automatically recalculate target
  - Trivial to verify
    - Difficult to find, but easy to verify
      - Nonce is published in block

The system automatically recalculates the target difficulty every certain number of blocks (e.g., in Bitcoin, every 2016 blocks).

As more miners join the network and computing power increases, puzzles must become harder so that blocks aren't solved too quickly. And vice versa.

Hash puzzles



Hash puzzles

- Simple PoW
  - Bitcoin uses SHA-256 as hashing function
  - Find SHA-256 hash value of string “blockchain”, such that

```
SHA256("blockchain" + Nonce) = A hash that starts
with "000000"
```

  - Numeric-only nonce values

```
SHA256("blockchain0") = 0xbd4824d8ee63fc82392a64414
44166d22ed84eaa6dab11d4923075975acab938

SHA256("blockchain1") = 0xdb0b9c1cb5e9c680dfff74
82f1a8efad0e786f41b6b89a758fb26d9e223e0a10

SHA256("blockchain2") = 0x8f0532cd22055fb7599aa4
8f38501dcd46e61712ab49a02f840f5545830e9260

..
..

SHA256("blockchain10730895") = 0x000000ca1415e0b
ec568f6f605fcc83d18cac7a4e6c219a957c10c6879d67587
```

Hash puzzles

- Simple PoW
  - Increase target by one additional leading zero (i.e., "0000000")

```
SHA256("blockchain934224174") = 0x0000000e2ae7e4240
df80692b7e586ea7a977eacbd031819d0e603257edb3a81
```

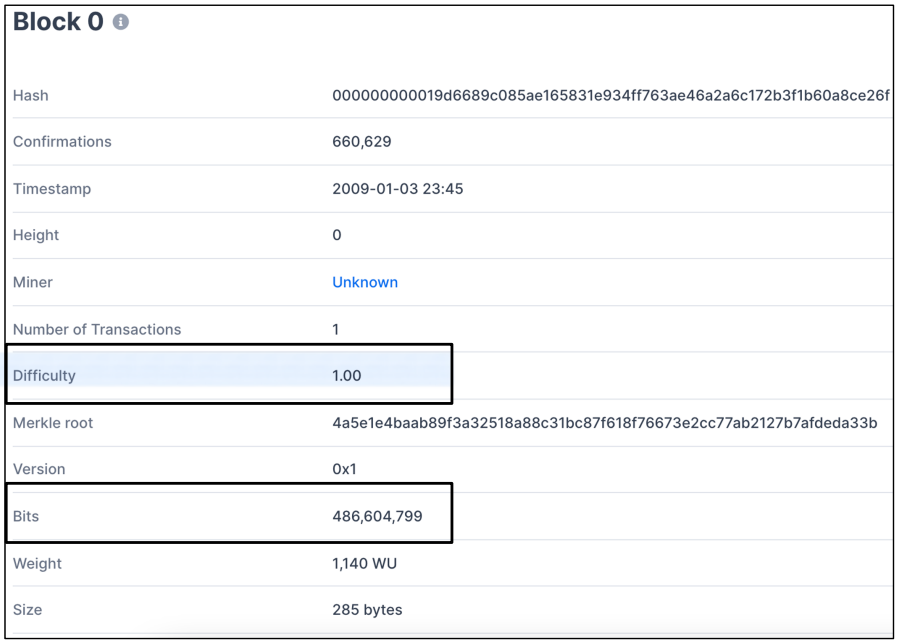
  - Additional leading zero
    - Overall target value is smaller than before
    - Increases the difficulty
  - Higher target value => more solutions exist => less difficult puzzle
    - No known shortcut to find a solution
  - Verifying solution is simple, single hashing operation

Hash puzzles

- Difficulty
  - How difficult it is to find a hash satisfying target value?
  - How difficult it is to find a block now
    - Relative to how difficult it would be at the highest possible target
      - Highest target = Lowest difficulty
    - If current difficulty = 6,695,826
      - With current hash rate, it is, on avg, ~6.6 million times as long/difficult to find a valid block as it would at a difficulty of 1

Hash puzzles

- Difficulty
  - Block# 0 (genesis block)



Block 0

Hash	000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f
Confirmations	660,629
Timestamp	2009-01-03 23:45
Height	0
Miner	Unknown
Number of Transactions	1
Difficulty	1.00
Merkle root	4a5e1e4baab89f3a32518a88c31bc87f618f76673e2cc77ab2127b7afdeda33b
Version	0x1
Bits	486,604,799
Weight	1,140 WU
Size	285 bytes

Hash puzzles

- Difficulty
  - Block# 0 (genesis block)
    - Difficulty = 1 **[Lowest difficulty]**
    - Bits = 486604799 = 0x1D00FFFF (compact format for target)

"Bits" is a compact representation of the mining target difficulty in Bitcoin, indicating how hard it is to find a valid block hash.

    - Target = 0x**0000 0000 FFFF** 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 **[Largest possible target]**
    - Hash = 0x**0000 0000 0019** D668 9C08 5AE1 6583 1E93 4FF7 63AE 46A2 A6C1 72B3 F1B6 0A8C E26F

- 256 bits = 32 bytes (count bytes in target/hash above)
- 0x00 –> 1 byte

Hash puzzles

- Difficulty
  - Block# 540277
    - Difficulty = 6,727,225,469,722.53
    - Bits = 388,618,029 = 0x1729D72D [1 byte index + 3 bytes coefficient]

1 byte is 2 hexadecimal bits.
So 1 hex bit is 4 binary bits.

    - Index = 0x17
    - Coefficient = 0x29D72D

    - Target = Coefficient * 2^(8 * (Index — 3))
      = 0x29d72d * 2^(0x08 * (0x17–0x03))
      = 2742061 * 2 ^(8 * (23–3))
      = 4.007526641161212986792514236082 8e+54
      = 0x0000 0000 0000 0000 0029 D72D 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Hash puzzles

don't know if this particular slide is important or not.

- Difficulty
  - Block# 540277
    - Target = 0X**0000 0000 0000 0000 0029 D72D** 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
    - Hash = 0X**0000 0000 0000 0000 000C 34C1** 851B 84F2 0F30 0CEA 21E5 6E32 FF2C BC80 F5C2 BCA3

    - Index = 0x17
    - Coefficient = 0x29D72D

    - Index (length of target) = **0x17 (23 bytes)**
    - Target = {(32-23)0x00} {**0x29D72D**} {(23-3)0x00}
    - Target = 0x0000 0000 0000 0000 00**29 D72D** 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000

Hash puzzles

- Difficulty

`diff_next` (Difficulty): A measure of how hard it is to mine a new block, increasing as more computational power joins the network.
`target_new` (Target): The maximum value a block's hash can be for the block to be valid, which decreases as difficulty increases.

  - Difficulty is adjusted every 2016 blocks (~2 weeks)

$$diff_{\text{next}} = (diff_{\text{prev}} * 2016 * 10) / (actual\_time_{\text{last\_2016\_blocks}})$$

```
target_new = difficulty_1_target / difficulty_new
```

  - Avg. time between blocks ~10 minutes

`difficulty_1_target` is the maximum possible target value, representing the lowest level of mining difficulty in Bitcoin.

  - Reflect overall computing power in the system
    - If last 2016 blocks were mined in > two weeks time
      - Overall computing power has decreased
        - Difficulty to solve PoW should decreased

- difficulty_1_target = 0x1D00FFFF

Hash puzzles

- Difficulty
  - New difficulty level is maintained at same value until adjusted
    - [0000-2015] => difficulty_x
    - [2016-4031] => difficulty_y
    - [4032-6048] => difficulty_z …
    - E.g., Block# 53869 or # 538696 or # 538697
      - Difficulty = 6,727,225,469,722.53
      - Bits = 388,618,029
  - First difficulty change (1 => 1.18) on block #32256 (2016 x 16)
  - Say, you are on #4035
    - Last block when difficulty changed = 4035 - (4035%2016)
      = 4035 - 3 = 4032 block

Hash puzzles

- Difficulty
  - Given target_current, compute difficulty_current?

→ formula for "diff new" is two slides above.

```
target_new = difficulty_1_target / difficulty_new
```

```
target_current = difficulty_1_target / difficulty_current
```

```
difficulty_current = difficulty_1_target / target_current
```

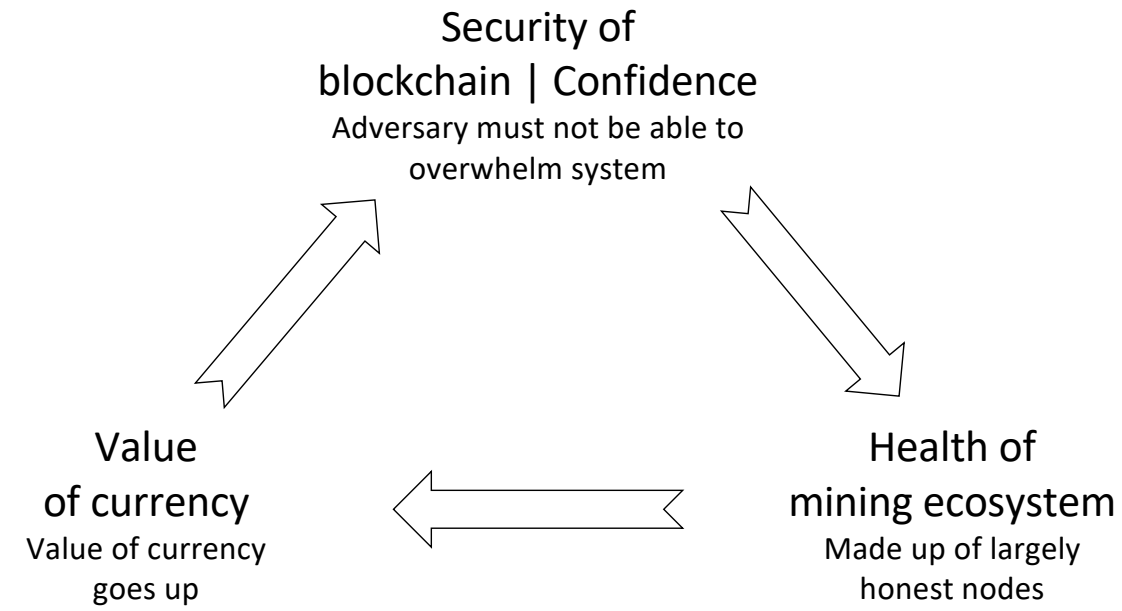
- Assignment: Derive difficulty+target+bits for current level
  - Go to prev change; go to prev-prev change (get/copy difficulty); compute forward for next level (compare results | show steps)

Hash puzzles

- Difficulty
  - Nothing magical about “10” (even “5” would be fine)
    - Just not too short (inefficiency, optimization of Tx in block)
  - A computationally difficult puzzle also helps combating sybil attack
    - Shifts focus of influence from identities to computational power

Bootstrapping

- Getting a cryptocurrency off the ground



- Bitcoin: No clear answer, partly media attention