Bitcoin and Nakamoto Consensus

Dhrubajyoti Ghosh

Introduction

- Although Bitcoin is quite unstable and it proof-of-work method is environmentally unsustainable, the Nakamoto Consensus protocol is in itself fascinating enough to warrant close introspection.

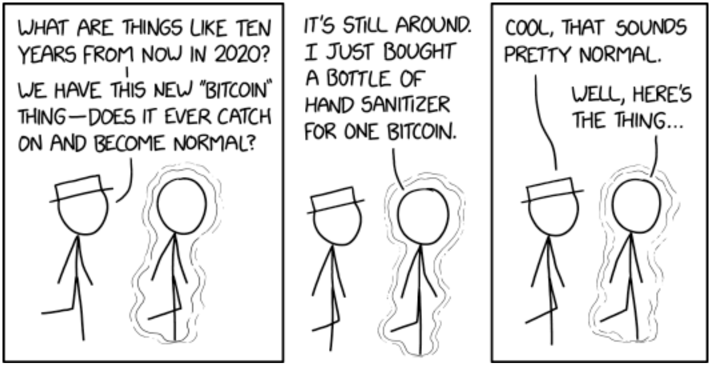

WHAT ARE THINGS LIKE TEN YEARS FROM NOW IN 2020?
WE HAVE THIS NEW "BITCOIN" THING—DOES IT EVER CATCH ON AND BECOME NORMAL?
IT'S STILL AROUND. I JUST BOUGHT A BOTTLE OF HAND SANITIZER FOR ONE BITCOIN.
COOL, THAT SOUNDS PRETTY NORMAL.
WELL, HERE'S THE THING...

Figure: 2010 and 2020, from XKCD

Introduction

- There is no notion of users "storing" bitcoins in some account.
- Rather, a public list of *all* transactions is maintained, and a user points to all the previous transactions in which he was involved, from which it is determined if he has enough to initiate a transaction.

The Problem

- Have network of nodes wanting to conduct transactions in the absence of a centralized bank.
- Spenders have to reference their earlier transactions when they want to make a new transaction.
- Malicious node A can reference the same transaction while making a payment to B and C.
- Each of B and C might not know that A is using the same transaction to pay the other, and might validate their respective transactions and send the product to A.
- In this way, A can successfully *double spend*.

The Problem (cont.)

- Double-spending cannot occur in a centralized bank.
- *How do nodes all agree on which transactions are valid?*
- Cannot emphasize enough: The network is *synchronized*, meaning that the time taken for messages to reach all nodes is bounded above by a known quantity.
- Other assumptions: permissionless system, meaning anyone can join and leave at any time.

Transactions (short)

- Say Alice wants to pay Bob 1 BTC. She broadcasts a message with this information, and *points to the previous transactions in which she was involved*.

- Alice also generates a *locking script* using Bob's public key. Anyone with Bob's private key can "unlock" this to avail the transaction.

- When Bob makes a transaction referencing Alice's transaction, he has to create an *unlocking script* with his public key and a digital signature to avail the amount.
  - Public key lets everyone know who is making the transaction.
  - Public key + Digital signature used to verify that Bob is indeed making the transaction (and not someone impersonating Bob).

- Digital signature generated using:- Bob's private key, ID of previous transaction, Alice's locking script, etc.

Transactions in detail

- Transaction has multiple inputs and multiple outputs (each output is for a different recipient).
- The contents of the transaction are also hashed (with SHA256 for e.g.) and the hash serves as an unique ID for the transaction.
- Each output contains a *locking script* called SCRIPTPUBKEY which needs to run successfully (in conjunction with SCRIPTSIG as defined below) for the recipient to redeem the amount.
- Each input of a transaction points to some earlier transaction by specifying the transaction hash (i.e. ID) and the index of the output which it would like to redeem. The input also contains an *unlocking script* called SCRIPTSIG.
- We now describe how SCRIPTPUBKEY and SCRIPTSIG work.

Transactions in detail (cont.)

- Consider two transactions $A \to B$ and $B \to C$. In the first transaction, A uses B's public key to create SCRIPTPUBKEY, which contains instructions that allow anyone with B's private key to redeem this transaction.
  In the second transaction, B creates SCRIPTSIG using
  - his public key (so the SCRIPTPUBKEY can match this with the key which A used), and
  - a digital signature generated using B's private key and details like the hash of the first transaction and the amount he plans to spend in the second transaction.
- If SCRIPTPUBKEY and SCRIPTSIG execute successfully together then the second transaction ($B \to C$) will be valid.
- No confidential information released in the process, hence transactions can be released freely in public.

Blockchain & Proof-Of-Work

- Each miner collects new transactions as they come into a block. On average, a new block is created every 10 minutes.

- The objective for the miner is to get everyone to agree on the transactions in his block (assuming they are valid transactions).

- Every block needs to point to a previous block (except the first block called the *genesis block*), whence the term *blockchain* originates.

- Different miners can work on different blocks.

Blockchain & Proof-Of-Work (cont.)

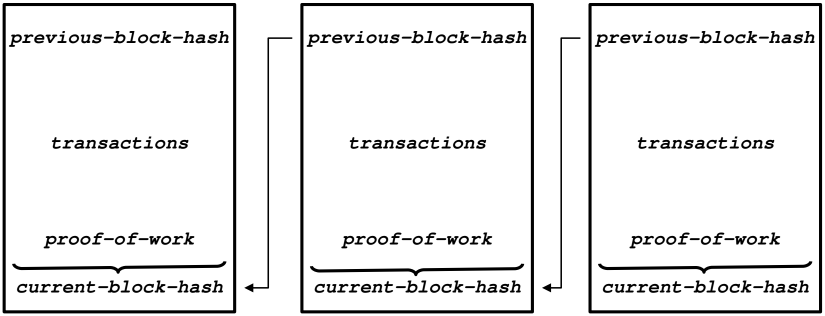


Figure: Blocks in blockchain

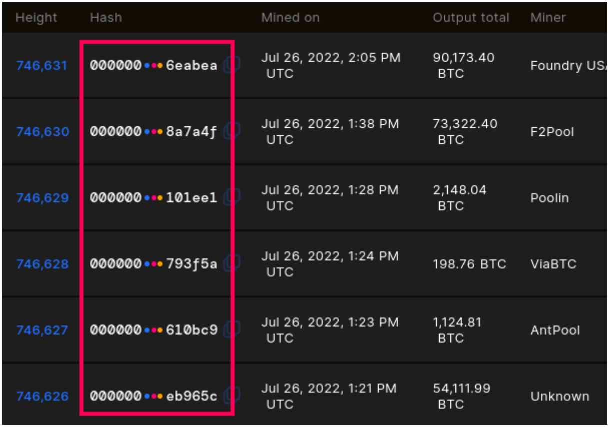
- Each block contains several data:
  - the hash of the *previous* block which the node wants the current block to point to (this is similar to maintaining a linked list),
  - list of transactions,

Blockchain & Proof-Of-Work (cont.)

  - a "proof-of-work" value (which we describe later), and
  - the hash *H* obtained by hashing the above data. This serves as the block's ID.

- "Proof-of-work" such that hash *H* starts with pre-determined number of 0s.
- To find proof-of-work, the miner working on the block keeps increasing a counter (called a *nonce*), checking if it satisfies the above.

Blockchain & Proof-Of-Work (cont.)



Height	Hash	Mined on	Output total	Miner
746,631	000000•••6eabea	Jul 26, 2022, 2:05 PM UTC	90,173.40 BTC	Foundry US
746,630	000000•••8a7a4f	Jul 26, 2022, 1:38 PM UTC	73,322.40 BTC	F2Pool
746,629	000000•••101ee1	Jul 26, 2022, 1:28 PM UTC	2,148.04 BTC	Poolin
746,628	000000•••793f5a	Jul 26, 2022, 1:24 PM UTC	198.76 BTC	ViaBTC
746,627	000000•••610bc9	Jul 26, 2022, 1:23 PM UTC	1,124.81 BTC	AntPool
746,626	000000•••eb965c	Jul 26, 2022, 1:21 PM UTC	54,111.99 BTC	Unknown

Figure: Notice the block hashes

Blockchain & Proof-Of-Work (cont.)

- Once the miner has a proof-of-work value, he broadcasts the block. Other nodes now easily check if the miner has indeed found a valid proof-of-work, and if the transactions in that block are valid.
- If they accept the block (they might not if they've already received a valid block from some other node), they express it by working on extending that block.
- Only transactions in the chain with highest total proof-of-work are considered valid i.e. only these can be referenced later; also nodes will ideally work on extending this chain. (More on this later)
- Usually this chain coincides with the longest chain.

Blockchain & Proof-Of-Work (cont.)

- *Thus to convince the other nodes to add one's block to their respective blockchains, one has to "solve" a difficult problem, namely the proof-of-work.*

  Remark: The number of 0s required required is constantly changing based on how easily the miners are able to find the proof-of-work for their blocks.

- *Secondary chains:* Nodes also keep track of blocks which form branches off the main chain.

- *Orphan blocks:* Valid block for which node cannot find a parent in its existing chains (can happen due to message delays).
  Saved in orphan block pool until parent is received.

- Above two needed in case some other chain grows longer, or node is working on wrong chain (due to message delay).

Significance of Proof-Of-Work

- Changing the contents of a block can change its hash, so if an attacker wishes to modify a transaction in some old block, he will again have to recalculate the proof-of-work for that block.
- But he has to add more blocks to his chain in order to overtake the current ideal blockchain.
    - Note that blocks which originally followed the modified block cannot be used as the hash of the modified block has changed.
- This requires calculating even more new proof-of-work values in a short period of time.
- Shall see that the probability that the attacker can convince the other nodes to switch to working on his version of the blockchain **reduces exponentially** with respect to the number of blocks that the attacker has to add.

Significance of Proof-Of-Work (cont.)

- *Main takeaway:* Finding proof-of-work is computationally expensive, so it is not in the interests of a node to find a proof-of-work for a block unless it is working on extending what it believes to be the ideal blockchain.

Why the chain with highest PoW?

- Have seen the existence of secondary chains, why in particular is the chain with associated highest PoW important?
- For brevity, call such chains *ideal chains*.
- Nodes consider transactions in the ideal chain to be valid (i.e. only these transactions can be referenced later), and work on extending it.
- *Why are only the transactions on ideal chain considered valid?*
- Consider the following scenario: A pays B for some product, B sends the product, but then A creates a fork by mining blocks not containing $A \to B$ so that they start before the block containing $A \to B$.
- Validating transactions in A's fork will be disastrous for B!

Why the chain with highest PoW? (cont.)

- As it is known that *A* has negligible chances of catching up with the ideal chain, only transactions on this chain should be valid.
- Will introduce the *n-deep confirmation* rule, which gives *B* more security from *A* launching double-spending attacks.
- *What is the incentive for miners to work on extending the ideal chain?*
- The incentive is *coinbase transactions*. These are special transactions which a miner makes to himself while mining a block.
- If that block is in the ideal chain, the miner can redeem the amount from this block.
- Hence it is in the interests of the node to extend the ideal chain.
- *Note:* Coinbase transaction introduces new currency into the system.

Double Spending attack

- Suppose a block containing the transaction $A \to B$ is added to the ideal chain.

- B will wait to send his product to A until there are at least n blocks (n is 6 in bitcoin) after the block containing $A \to B$, and if this block is still in the ideal chain.

- Sometimes called the *n-deep confirmation rule*.

- In order for A to successfully double spend, A will try to fork the chain from the block preceding the block containing $A \to B$, with no mention of this transaction.

Double Spending attack (cont.)

- A creates the blocks in secret until B sends his product. If A's chain now contains more blocks than the current ideal chain containing $A \rightarrow B$, she broadcasts the blocks in her chain.
- The transactions in A's chain become valid and the transaction $A \rightarrow B$ is invalidated. A can now reuse the amount she had otherwise paid to B.
- *Note:* Note that if A broadcast her blocks before n confirming blocks were added to the current chain, then B would cancel his product shipment.

Another possible attack

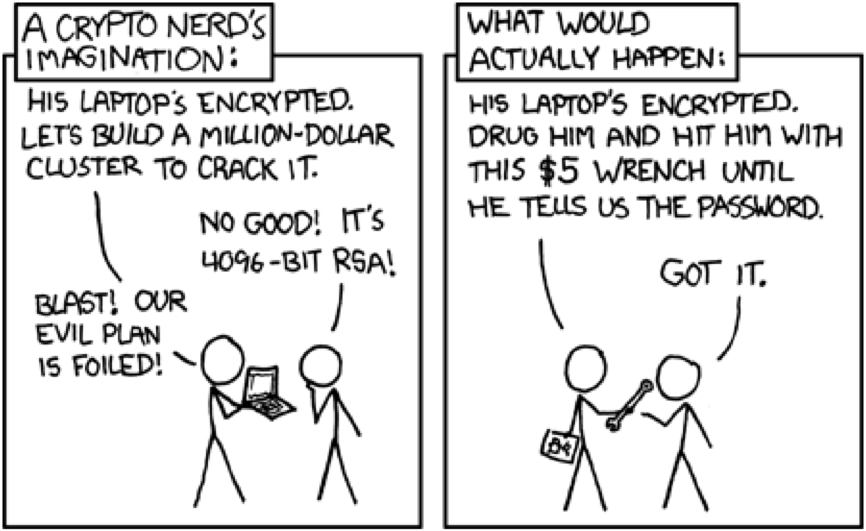

A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Figure: Security, from XKCD

Incentives

- Have covered the incentive of coinbase transactions.
- *Transaction fees* are another incentive.
- Incentivize miners to include transactions into their blocks and discourage spam transactions.
- Encourages nodes to not carry out double-spend attacks.
- If a node has enough power to overtake the ideal chain, it can use the power to instead mine blocks and add them to the ideal chain.
- Total transaction fees availed might be more than what it would gain from double-spend attack.

Working on the same blockchain

- Essential that all nodes agree on the same blockchain in order to agree on which transactions are valid.
    - For e.g. B might ship his product based on the transaction $A \to B$ which he believes is in the ideal chain but is actually not.

- Most nodes (i.e. honest ones) will want to work on ideal chain, so will be more-or-less in consensus.

- *Not-uncommon situation:* A and B find the proof-of-work for their blocks (pointing to the same previous block) in quick succession and broadcast them before either of them is aware of the other having solved the proof-of-work.

- If time gap is larger, most nodes will already settle on the block they receive first, so no problem in this case.

Working on the same blockchain (cont.)

- Due to small time gap between *A*'s and *B*'s blocks, nodes do not know which chain to extend.

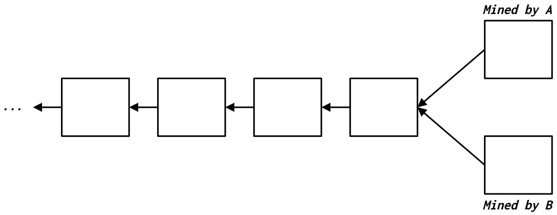


Figure: A fork

- Maybe the next block will arrive in isolation, say it extends *A*. Then all nodes converge on *A*'s fork.

Working on the same blockchain (cont.)

- If however each fork is again extended by one block in a short period (albeit with small probability), wait for the above to happen.

- *What if a node works on a chain that is not the ideal chain, due to message delays, Byzantine faults, etc.?*

- The network is *synchronous*, and after some time the node will notice that it is not working on the ideal chain. Unless he is sure he can overtake the ideal chain, he'll switch over to avoid wasting power.

- *Suppose A and B find their blocks in a small time gap, but A has received B's block before transmitting his own. Should A abandon trying to extend his block even though he has worked hard to get it?*

- For all *A* knows, all other nodes will have received *B*'s block and started extending it, so they will probably not work on *A*'s block.

Calculations

- We shall assume the following:
    - No two nodes find a block at exactly the same time.
    - The total hashrate of the honest network and the attacker is constant, say H. Let the honest network have hashrate pH while the attacker has qH hashrate, $p + q = 1$.
    - Mining difficulty is constant, and with hashrate H the average time to mine a block is T_0.
- Also, the hashing function is assumed to be a random oracle, meaning that the outputs of two different inputs are independent.
- It is intuitively clear that if the honest network and attacker start mining their blocks at the same time, the probability that the former (resp. latter) mines a block first is p (resp. q). `Formal proof in Appendix`

Catching up

- Will assume that ideal chain and longest chain are always *the same*.
- Suppose the honest network has already built n blocks on top of the block preceding the block with $A \to B$ while the attacker has built m blocks.
- Let $z = n - m$. Want to find probability that z will become -1 (i.e. attacker will overtake honest network) at some time.

Catching up (cont.)

Lemma

$$a_0 = \begin{cases} 1, & \text{if } p \leq q \\ q/p, & \text{if } p > q \end{cases}$$

Proof.

- Interpret process of mining blocks as a sequence $\{z_1, \cdots\}$ where z_i is -1 or $+1$ depending on whether the i-th node to mine a block is the attacker or the honest network.
- Odd number of steps required to go from $z = 0$ to $z = -1$ due to parity reasons.
- a_0 is the probability of getting a sequence $\{z_0, z_1, \cdots, z_{2k+1}\}$ (for some $k \geq 0$), so that sums of all prefixes are ≥ 0 except for the last, which is -1.

Catching up (cont.)

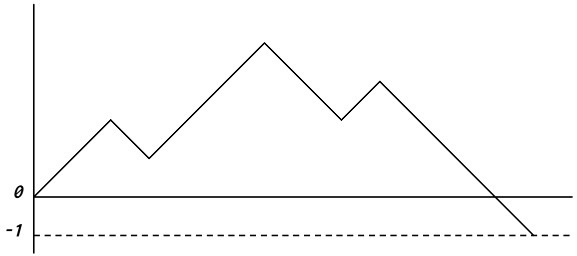


Figure: Random walk from $z = 0$ to $z = -1$

- The number of such sequences of length $2k + 1$ is equal to the k-th Catalan number C_k.

Catching up (cont.)

- Also, the *generating function* for the Catalan numbers is

$$\sum_{k\geq 0} C_k x^k = \frac{1-\sqrt{1-4x}}{2x}.$$

- Probability of getting such a sequence is thus

$$\begin{aligned}\sum_{k\geq 0} C_k p^k q^{k+1} &= q\sum_{k\geq 0} C_k (pq)^k \\ &= q\cdot\frac{1-\sqrt{1-4pq}}{2pq} \\ &= \begin{cases} 1, & p\leq q, \\ q/p, & p>q \end{cases}\end{aligned}$$

Q.E.D.

Catching up (cont.)

- Clearly $a_x = -1$ for $x < 0$.
- $a_x = a_0^{x+1}$ for $x \geq 0$. Going from $z = x$ to $z = -1$ same as going from $z = x$ to $z = x - 1$, then from $z = x - 1$ to $z = x - 2$ and so on.
- Thus

$$a_x = \begin{cases} 1, & p \leq q, \\ (q/p)^{x+1}, & p > q. \end{cases}$$

Overriding n-deep confirmation

- Goal is to calculate probability of B reading $z = -1$ (i.e. overtaking A) if A follows n-deep confirmation rule.
- Let $P(m)$ denote the probability of attacker mining m blocks before honest network mines n blocks. Then

$$P(m) = \binom{m+n-1}{m} p^n q^m.$$

- Assuming $p > q$, required probability is

$$\sum_{m \geq 0} P(m) a_{n-m} = \sum_{m=0}^{n} \binom{m+n-1}{m} p^{m-1} q^{n+1} + \sum_{m>n} \binom{m+n-1}{m} p^n q^m$$

- If $p \leq q$, required probability is 1.

Analysis

- Protocol is probabilistic, double spending can happen even if $p > q$.
- If attacker has majority hashrate, no amount of confirmations can stop double-spending attacks.

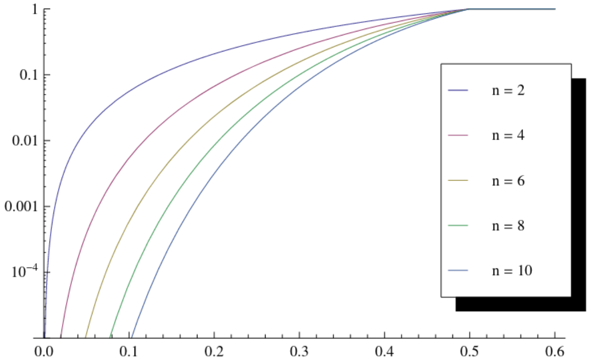


Figure: Chances of double-spend as $q \to 0.5$ for different n

Synchronization, etc.

- Most papers assume that the network is synchronous.
  - Nakamoto's white paper
  - Meni Rosenfeld: *Analysis of hashrate-based double-spending*
  - Garay, Kiayias, Leonardo: *The Bitcoin Backbone Protocol: Analysis and Applications*
  - Ling Ren: *Analysis of Nakamoto Consensus*
- Pass, Seeman and shelat consider the protocol in asynchronous networks; *Analysis of the Blockchain Protocol in Asynchronous Networks*

Unresolved doubts

- Nakamoto has mentioned that PoW can be used to solve the Byzantine General's problem. But under what assumptions?

Appendix

Proposition

If honest network and attacker have pH and qH hashrate, probability of finding a block by the honest network (resp. attacker) equal to p (resp. q).

- Hash values for different inputs are independent of each other.
- The time taken to find a PoW for a block has no bearing on the time taken to find the next PoW of a block.
- Due to this *memorylessness*, block mining treated as Poisson Process .
- Informally, in a Poisson process $\{N(t) \mid t \geq 0\}$ with rate λ,
  - $N(t)$ is random variable counting number of events in $[0, t]$.
  - interarrival times are i.i.d. with exponential distribution with parameter λ.

Appendix (cont.)

- Let $N_1(t)$ with rate λ_1 and $N_2(t)$ with rate λ_2 be Poisson processes corresponding to honest network and attacker.

- Processes $N_1(t)$ and $N_2(t)$ **are independent** as attacker and honest network have different blocks (due to different coinbase transactions).

- Then $N_1(t) + N_2(t)$ is a Poisson process with rate $\lambda_1 + \lambda_2$ and probability that the first event of this process comes from $N_1(t)$ is $\frac{\lambda_1}{\lambda_1+\lambda_2}$.
  - This is Theorem 8.12 in Probability and Computing by Mitzenmacher and Upfal.

- Assumed that with hashrate H, expected time to mine block is some T_0.

Appendix (cont.)

- Then expected time for honest network (resp. attacker) to mine block is T_0/p (resp. T_0/q).

- Since the inter-arrival times have exponential distributions (where mean is reciprocal of rate), $\lambda_1 = p/T_0$ and $\lambda_2 = q/T_0$.

- Probability that honest network mines a block first is

$$\frac{\lambda_1}{\lambda_1 + \lambda_2} = \frac{p/T_0}{p/T_0 + q/T_0} = p.$$

References and Further Reading

- Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*
- Andreas Antonopoulos, *Mastering Bitcoin*
- Meni Rosenfeld, *Analysis of Hashrate-Based Double Spending*
- M. Mitzenmacher, E. Upfal, *Probability and Computing*
- Ling Ren, *Analysis of Nakamoto Consensus*
- J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, E. W. Felten, *SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies*
- R. Pass, L. Seeman, a. shelat, *Analysis of the Blockchain Protocol in Asynchronous Networks*

References and Further Reading (cont.)

- https://decentralizedthoughts.github.io/2021-10-15-Nakamoto-Consensus/
- https://github.com/decrypto-org/blockchain-papers
- https://courses.grainger.illinois.edu/ece598pv/sp2021/lectureslides2021/ECE_598_PV_course_notes3.pdf
- https://satoshi.nakamotoinstitute.org/emails/cryptography/11/