EE 374: Blockchain Foundations **Stanford, Spring 2022**

Lecture 15: From Proof-of-Work to Proof-of-Stake

May 20, 2022

Lecturer: Prof. David Tse Scribes: Gordon Chi, Albert

Reminder: Take-home test is June 7th, 2022.

1 Proof-of-Stake

1.1 Proof-of-Work

In previous lectures, we have seen two main motivations for using Proof-of-Work (PoW) in blockchain technologies. First, it is a defence against **Sybil Attacks**, since it forces participants to use computing resources to create new blocks. Second, PoW serves as an **arrow of time**: it ensures that blocks are not generated too fast (much faster than the network delay Δ), and hence ensures safety of the system (i.e., proof of work prevents confusion amongst honest parties within the network and makes double spending hard).

1.2 Drawbacks of PoW

Unfortunately, there are a few drawbacks of Proof-of-work.

- **High energy consumption**: As a motivating example, consider Bitcoin, which has a current hash rate of ≈ 200 exahashes (exa $= 10^{18}$) per second, i.e. the hash rate of Bitcoin is of the order of 10^{20} hashes per second around the world across all miners (Figure 1). The total energy consumption for computing so many hashes is close to that of Argentina, a country with over 45 million residents. While a significant percentage of that energy comes from renewable sources, it is still a significant consumer of energy.
- **Centralization**: In their original paper, Satoshi Nakamoto had envisioned decentralization as an advantage of blockchain technologies [2]. While anyone can still start their own miner, most of the hash rate comes from specialized mining facilities who have economies of scale. This has led the industry to become centralized to those who have the capital to build these facilities [1].

1.3 Proof-of-X: Alternatives to Proof-of-Work?

Proof-of-X refers to cryptographic protocols designed to replicate the important features of proof-of-work using resource X, without the aforementioned drawbacks. Most recently released blockchains use **Proof-of-Stake (PoS)**, where *one-CPU-one-vote* of a PoW protocol is replaced with *one-coin-one-vote* in the Proof-of-Stake protocol. PoS protocols strive to be resistant to Sybil Attacks by forcing an attacker to acquire a large portion of the blockchain's tokens.

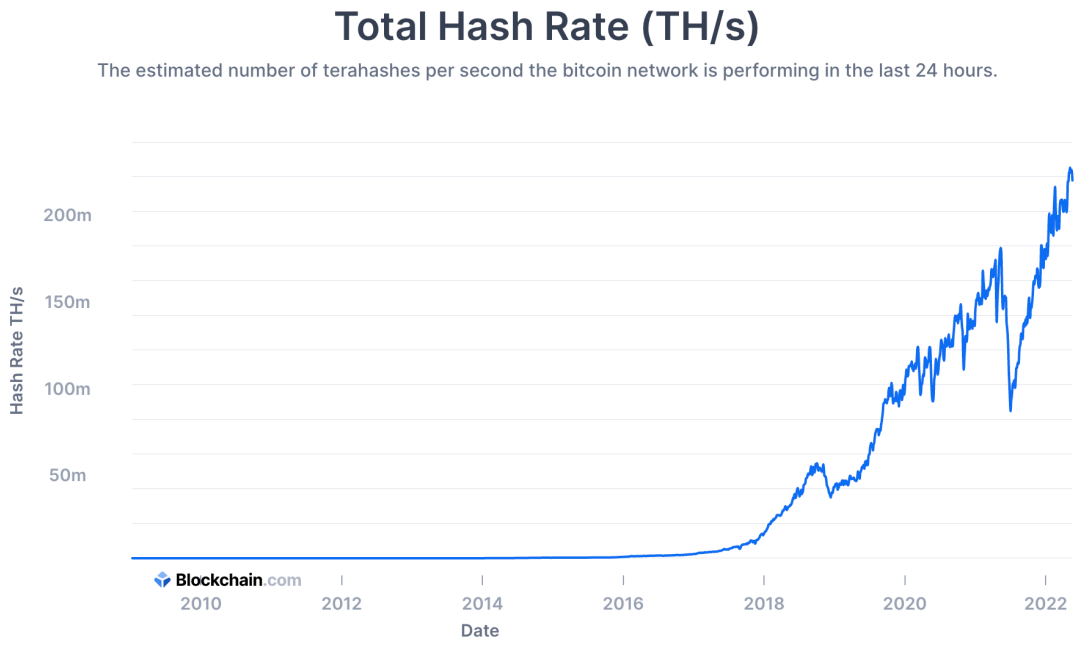

Figure 1: Total Hash Rate from 2010 to 2022 for Bitcoin.

1.4 New Attack Vectors

The introduction of Proof-of-Stake gives rise to new attack vectors we would otherwise not encounter in a Proof-of-Work protocol. This section will not go into detail on how we prevent these attacks, but rather give an intuition as to why these attacks happen:

- **Grinding Attacks:** PoW can be viewed as a random lottery to select a miner to propose a block. Many PoS protocols have similar lotteries to select a staker to propose or vote for a block. Grinding attack occurs when an adversary biases the randomness in her favour; thus effectively holding more votes in the network per coin. This results in an honest stake majority no longer being sufficient for security of the protocol.

  The name of this attack comes from how an adversary can "grind" through many possible seeds of randomness in order to increase the probability of generating a valid block.

- **Prediction Attack**: In Proof-of-Work, it is completely unpredictable who will get the next block. However, in Proof-of-Stake, this is no longer the case. We will see in this lecture how Proof-of-Stake leads to predictability. Predictability leads to attacks where the adversary can bribe the winner of the next block to carry out malicious actions. We will also see in the next lecture how cryptography can be used to ameliorate this problem to some extent.

- **History Revision Attacks:** In Proof-of-Work, it is impossible to rewrite the history of the blockchain over a long period of time since one cannot redo all the work done in a long period of time, a property of the arrow of time. However, since the concept of "work" does not exist in Proof-of-Stake, it is easier to rewrite history, thus opening up the possibility of such attacks.

While this all may seem a bit pessimistic, fear not...

1.5 Advantages of PoS

With danger comes opportunities! Here are some properties of consensus that are possible in Proof-of-Stake and are either not possible or hard to accomplish in Proof-of-Work:

- **Fast Confirmation:** Bitcoin requires a long confirmation time for transactions because one needs to wait for a block to be buried under k blocks (this process can take hours), whereas some Proof-of-Stake protocols can confirm transactions in seconds.
- **Accountability:** While Nakamoto's paper introduced the use of incentives in consensus, there is no punishment system for malicious actions since miners have no identity. On the other hand, in a one-coin-one-vote Proof-of-Stake system, we know who is participating in the system, so it is **possible** to take away a staker's stake if they violated the protocol.

2 Two Classes of Protocols

As a signpost for the next four lectures, these are the two main classes of protocols we will discuss.

- **Proof-of-Stake longest chain** (our discussion today and next lecture): This class of protocols follow the longest chain design but replaces PoW with PoS.
- **BFT (Byzantine Fault Tolerant)** (last two lectures): This class of protocols has the advantages of accountability and fast confirmation.

Ongoing research attempts to combine the advantages of both protocols into one, but is beyond the scope of this course.

3 Changes to the PoW Equation

In a proof-of-work protocol, the proof-of-work equation is designed such that there exists in expectation of the order of one block per round, with a round defined as the network delay. (As we know from previous lectures, we cannot ensure exactly one block per round).

We want to preserve a similar property in a proof-of-stake protocol. Thus, let us consider how our new PoS equation should be constructed, given that we have n stakers instead of n miners[1]. We recall that our proof-of-work equation is

$$H(s||x||\text{ctr}) < T \tag{1}$$

with T defined as the target variable, x as Merkle root of the list of transactions, and s as the hash of the previous block. What are the changes required for a proof-of-stake protocol? We go through a step-by-step process of this conversion.

[1] We may have changes in the number and set of stakers in a real life implementation, but for now let us fix them for a simpler model

3.1 Step 1: Replace the Nonce

The process of finding the nonce (defined as *ctr* in the proof-of-work equation) leads to the "work" required into the creation of a block, so our first step is to replace this parameter with the public key of a staker, i.e.

$$H(s||x||pk_i) < T_s.^2 \tag{2}$$

In this scenario, all stakers $i = 1, 2, \cdots, n$ will try this equation with their own public key, and whoever has a valid key will be able to propose the block, winning the "lottery". For now, we will use a simplifying assumption that each staker chooses their public key in advance and does not grind on the public key.

However, this modification to our equation is not sufficient as is. Most notably, consider a fixed round k and suppose that with the fixed n stakers, there exists no $i \in \mathbb{N}, 1 \leq i \leq n$ such that pk_i satisfies $H(s||x||pk_i) < T_s$. Then, nobody wins this "lottery", and the chain is not live anymore, a violation of an important ledger property. If our new proof-of-stake equation fails at any given round r, we want there to be some non-zero probability that a staker can fulfill the equation at a later round $r + 1, r + 2, \cdots$. This thus leads us to the next step.

3.2 Step 2: Add the round index r

Our PoS equation is now defined as

$$H(s||x||pk_i||r) < T_s \tag{3}$$

and we run this for every round r and i until the statement succeeds for a given staker and a block is constructed. We keep blocks from future rounds in the buffer until that round arrives, and ensure that blocks along a valid chain have strictly increasing round numbers.

Again, however, there are caveats with the equation as is. Considering the attack vectors in Section 1.4, we observe that our equation is still vulnerable to grinding attacks. Specifically x is a grinding opportunity as the adversary can try many different transactions as well as ordering of the transactions to fulfill the equation given a fixed previous hash s, public key pk_i and round number r.

This means that a computationally powerful adversary can greatly increase her chance of winning a block, thus resulting in our protocol acting like a proof-of-work protocol. We cannot have that in PoS, so x must be removed.

3.3 Step 3: Remove x

Our PoS equation now looks like

$$H(s||pk_i||r) < T_s \tag{4}$$

and we have the **winning staker sign the set of transactions** that they include in the block. This may be a little vague, but this concept will be discussed in the next lecture.

As a motivation for the signing step, we consider the ramifications of removing x from the equation: x was a crucial piece in the PoW puzzle because it prevented the tampering of transactions

[2] the T_s value here denotes that we may be considering a different target value from the original Proof-of-Work equation since we have a Proof-of-Stake protocol.

in the block. Without x anywhere in our block, we are left with a scenario where stakers of the blockchain can tamper with the transactions in a block and still satisfy the proof-of-stake equation. The signature prevents nodes in the network from tampering with the transactions as they cannot forge the signature. The current system (with the signature) still does not prevent an adversarial winning staker from changing the transactions in the block and re-signing the block, but this is something to be explained in the next lecture.

We are almost done with our PoS protocol, but once again the equation still has a problem: s remains a grinding opportunity: an adversarial party can try to attach to shorter chains in the hope of getting more favourable chances of winning blocks in the future. This thus results in the final change we make today.

3.4 Step 4: Change the previous hash s

Since we identified that s is a grinding opportunity, we replace the hash s in our equation with

$$s_0 = H(G) \tag{5}$$

where $H(G)$ is the hash of the genesis block. which thus yields us

$$H(s_0||pk_i||r) < T_s. \tag{6}$$

Now, we have successfully constructed a proof-of-stake equation where there are no longer any grinding attacks possible (since s_0, pk_i and r are fixed and cannot be grinded)! However, in the next lecture, we will cover the disadvantages of our current conclusion, namely, this system is highly predictable (as s_0, pk_i and r are known in advance). We will see how prediction attacks can occur and how to fix the problem of prediction.

References

[1] A. Beikverdi and J. Song. Trend of centralization in bitcoin's distributed network. In *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 1–6, 2015.

[2] S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. `https://bitcoin.org/bitcoin.pdf`.