EE 374: Fundamentals of Blockchain Infrastructure **Stanford, Spring 2025**

Lecture 12: Long Range Attacks and Light Clients

May 7, 2025

Lecturer: Prof. David Tse Scribe: Jeremy Kim

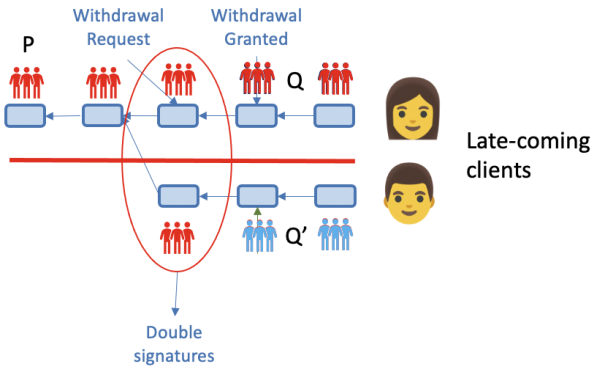
1 Validator sets

In previous lectures, we have assumed that the validator set stays fixed. However, in practice, as the chain grows, some validators may want to leave and retrieve their stake. When the chain grants this withdrawal, validators still have their keys, so they can sign on heights where those keys were valid and possibly create forks, which we call a **long-range attack**.

This is a problem for late-joining clients. Consider the following:

1. $\frac{2}{3}$ of the validators leave and then start forking. Remember that safety is only guaranteed when at most 33% of the validators are malicious, so in this situation, a fork can be created
2. Late joiners cannot tell which fork to accept
3. Economic security would lead to clients identifying 1/3 of the validators as malicious, but those entities cannot be slashed because they already withdrew

The diagram below visualizes the long-range attack. The validators in the circled area create a withdrawal request. Once they successfully retrieve their stake, they go back to the height where their keys are accepted (the bottom chain) and create a fork.



How do you resolve this issue? One solution is social consensus and another leverages Bitcoin. After discussing these two ideas, we will discuss scaling blockchains.

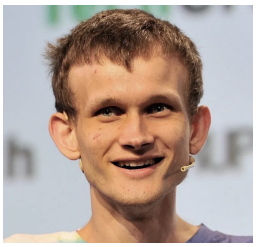
2 Social consensus

A fork in a proof of stake chain can be discarded if clients know which branch was established first. **Therefore, we can make the withdrawal time long.** Then, some clients can tell others which fork is correct and establish **social consensus**.

Let's consider the following example to motivate social consensus. Ethereum caps 16 withdrawals per block. In order for the aforementioned attack to succeed, out of the current 1 million validators (approximate), an order of 700,000 withdrawals would need to occur. 1 block is validated every 12 seconds, so it would take 1 week for all of these validators to exit and then mount their attack. In this scenario, social consensus asserts that during that week, Ethereum participants can agree on which part of the chain to accept in the event of a fork.

Ethereum is not the only chain that relies on social consensus. In the Cosmos chain, the unbonding period is 21 days. If a client comes online once every 21 days, then they can identify which branch was created earlier in the event of a fork.

Note that in this current setup, a client needs to be online consistently. What should clients do when they can't be online as often? They can trust one entity/person to continuously supervise the blockchain. However, this is not ideal, so it motivates the next solution which uses Bitcoin.



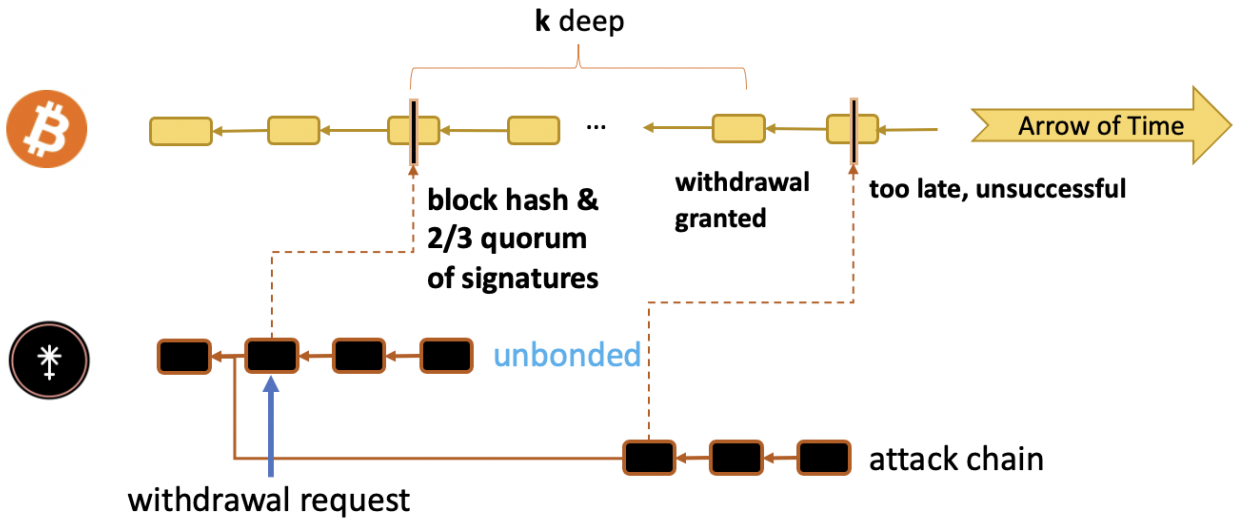
Ask Vitalik Buterin about forks
Image source: Decrypt.co

3 Bitcoin as a timestamp server

A trusted superviser effectively timestamps a proof of stake chain's events. We can make Bitcoin a trusted timestamp server. An illustrative example is below.

Suppose you wanted to prove that you found a new discovery at 3:50 PM on May 7, 2025. You are worried that someone will take your discovery in the next 24 hours. Here is one solution: find a way to hash your work and then create a Bitcoin transaction containing that hash. The transaction will be validated within the next 10 minutes (approximately), so you can establish a definitive time of discovery.

A similar idea can defend against long-range attacks. When a withdrawal request is made, a node can hash that transaction (including the signatures for that block) and publish it as a Bitcoin block. Within the next few hours, the Bitcoin block will be k-deep, after which the proof of stake chain can approve the withdrawal request. When clients encounter forks, they can choose the branch which contains the earliest timestamped block (rejecting branches where timestamps cannot be found). By Bitcoin's security, malicious validators cannot obtain an earlier timestamp than the time at which they withdrew their stake. This technique is trustless!



As a technical note, this method is implemented by establishing timestamps in two Bitcoin transactions, costing just over a dollar in total. Specifically, the BLS signature scheme lets you create a 128 byte signature that aggregates thousands of signatures into one signature. We can create a signature of all the valisdators who signed the withdrawal request to snapshot the active validators. This signature fits in two 80-byte `OP_RETURNS`.

4 Blockchain scaling

Bitcoin's throughput is 7 transactions per second (tps); Ethereum's is 20 tps. Ideally, we can increase this throughput without affecting security and decentralization.

1. Can you increase block size? If you increase the block size, it takes more time to process blocks and latency goes up, increasing the forking rate and decreasing security
2. Can you give nodes more compute? Throughput will go up, but it decreases decentralization

Instead, lets focus on increasing decentralization and keeping the throughput the same. Recognize that nodes do not need to download the full chain. Instead, they often only need to look at block headers and process blocks that they deem relevant based on this metadata.

4.1 Light Client

We will start our upcoming discussion on blockchain scaling with this description of a light client. Specifically, this client can check a transaction's validity without downloading the entire chain. The client works as follows:

1. Download all headers from a full network node.
2. Verify that proof of work has been done properly based on the hashes of previous blocks (found in the block metadata).

3. Confirm that the chain you are working with is indeed the longest chain. This issue is less of an issue on Tendermint, where blocks can be finalized by a certificate (in the absence of a long-range attack). On Bitcoin though, you can't directly check that the chain is the longest chain. This can be resolved by asking multiple nodes (i.e. 6 nodes). With some probability, you will contact at least 1 honest node. Then the honest node will give you the longest chain.

4. To verify whether a block contains a transaction, you can look at each its header's Merkle root field. This value is the root of a Merkle tree representation of that block's transactions. If a prover wants to prove to a client that a transaction was included in a block, it can provide the client with the transaction's Merkle path. The client can hash the transaction with the Merkle path and confirm inclusion if the resulting hash value matches the Merkle root value. Please see the visual below for more details.

5. The transaction is valid if a block containing that transaction was included in the longest chain.