ECE 595: Foundations of Blockchain Systems **November 18, 2020**

Lecture 14: Proof-of-Stake

Lecturer: Sreeram Kannan *Scribe: Haorui Ji*

Outline: This lecture first specifically address the problem of "Nothing at Stake Attack". Then we introduced ideas from the Ouroboros protocol aiming to minimize the nothing-at-stake attack. Finally we start the discussion of Nakamoto block which aims to close the gap of system security performance when the adversary fraction β is in the interval $\frac{1}{3} \leq \beta \leq \frac{1}{2}$.

14.1 Recap

In the last lecture, we looked at several alternatives to proof-of-work mechanism, like proof-of-stake, proof-of-space, etc. and we focused on the proof-of-stake(PoS). The core concept in PoS is the leadership certificate(L.C.), which is required to create a new block:

$$H(Prev.L.C., pk, t) < threshold * stake(pk) \tag{14.1}$$

14.1.1 Equivalent Question

One way of thinking about the equivalence of leadership certificate if that once we're given the previous L.C. and a public key, can we calculate the earliest time t_0 that we can mine a new block:

$$given\ \{Prev.L.C., pk\} \rightarrow earliest\ time\ t_0\ to\ mine\ a\ new\ block \tag{14.2}$$

There are two major differences in these two equivalence:

1. Eq.14.1 assumes exponential interarrival process while Eq.14.2 assumes general interarrival process.

2. Eq.14.1 is more relative to a "mining" process where nodes try to mine one block at each time stamp. Eq.14.2 can immediately "know" when to get a new block from a single evaluation. Therefore, the second idea can be used for implementing PoS efficiently.

14.2 Nothing at Stake Attack

In here, we still assumes the ”static stake” prerequisite, where all public keys and their stake are defined in the genesis block and do not change.
While honest nodes mine only at tip of the longest chain, adversarial nodes can mine on all previous leadership certificates, i.e. instead of forming a chain, adversaries can form a tree structure and try to outrun the honest chain. This is called the Nothing-at-stake attack.

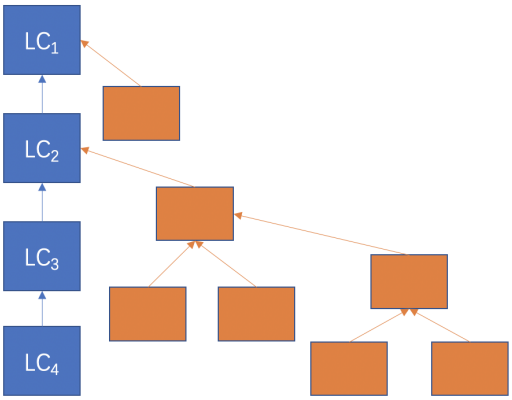

Figure 14.1: Nothing-at-stake attack

Note that in Proof-of-Work mechanism, if you want to mine blocks simultaneously on other blocks, you'll have to split your mining power between them. But, in Proof-of-Stake case, you will not splitting your stake since stake is already defined at genesis. For example, if you had 30% stake in total, you'll have 30% stake on every block you mined on therefore you can mine multiple blocks simultaneously. In addition, adversaries benefit greatly from the independent randomness of mining anywhere on their tree, so they are highly likely to grow faster than the honest chain.

14.3 Ouroboros / Snowwhite Protocol

These protocols have very similar properties and they are all aim to minimize the Nothing-at-stake attack.

14.3.1 Specify Behaviors of Different Nodes

Recall that Eq.14.1 define the equation that all nodes need to follow. Different nodes tend to behave different at time stamp t under this protocol. For honest nodes, they only try to achieve Eq.14.1 with $Prev.L.C.$ corrsponding to the tip of the longest chain. However, for adversarial nodes, they try to achieve Eq.14.1 with all $Prev.L.C.$ as all previous blocks to see how they can maximize the chance of creating a new block.

14.3.2 Idea-1

The main idea is to reject blocks whose time is too old relative to current block:

$$t(block) < t(clock) + \gamma \rightarrow Rejection \tag{14.1}$$

Issues with this idea:

1. The parameter γ is highly dependent on the assumption of the rate of chain growth. However, this is not a big deal since we can always have a good estimation of the rate of chain growth or we can regulate that rate.

2. A much more severe problem is the "divergent view" problem. For example, an adversary can have time a block at the boundary so that half of the honest nodes will accept this block while the other half will deny it. Therefore, these two half of nodes will have different view of longest chain and will never agree on each other. An extreme scenario of this is called "network partition":

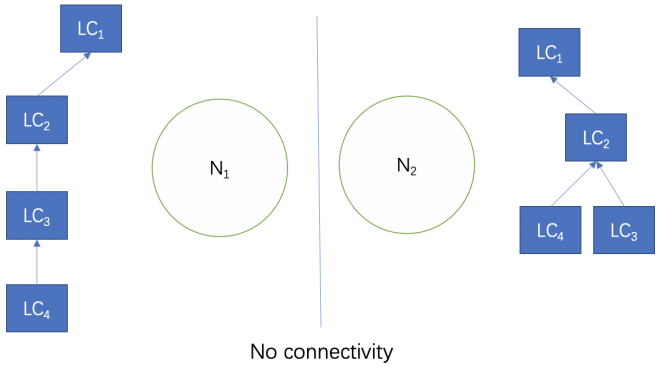

Figure 14.2: Network Partitioning

Honest nodes are partitioned into two groups N_1 and N_2 with no connectivity, and both of them will have its own view of chains. After network mixes, if it were Bitcoin protocol, everybody will accept the longest chain, but in here, two parts will not agree on each other and therefore split forever. Therefore, this idea is not robust to network asynchrony(partition).

14.3.3 Idea-2

The second idea is to just simply remove the $Prev.L.C.$ term in the mining equation:

$$H(pk, t) < threshold * stake(pk) \tag{14.2}$$

In previous Eq.14.1, when we have $Prev.L.C.$ we can only attach current L.C. to the previous leadership certificate, but now we can attach the leadership certificate anywhere.

14.3.3.1 Validity of Leadership Certificate

1. In the L.C. chain, time stamps should be monotonically increasing.
2. The timestamp should be earlier than time of current clock
3. Hash condition satisfied

Recall the three-layer structure of PoS protocol, L.C can be floating independently, but header chain and block chain can still connect to each other and form a chain structure. Therefore when we're talking about the L.C. chain, we're actually talking about the header chain and regard the two L.C. on these two connected headers as connected also.

14.3.3.2 Specify Nothing-at-stake Attack In This Case

Under this setup, this looks like a different Nothing-at-stake attack compared with what we have seen earlier.

1. For honest nodes: they grow a block at tip of the longest chain at time t
2. For adversary nodes: if they create a block at time t, they can attach it everywhere on the existing structure.

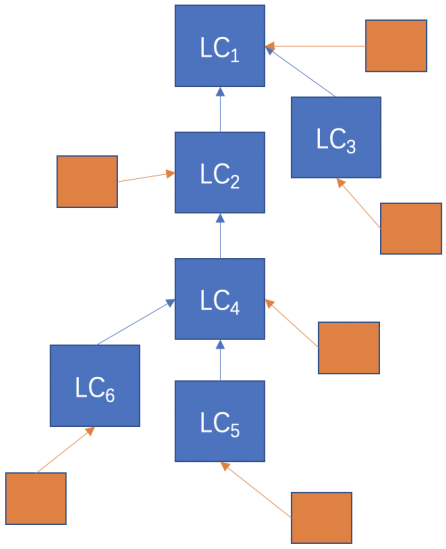

Figure 14.3: NaS attack

14.3.3.3 Full Proof

Full proof means we try to show security under all sorts of attacks. In here, we still assumes that we have network delay $\Delta = 0$

Suppose we have the following two chains:

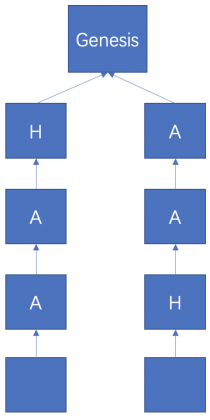

Figure 14.4: Two-chains Example

Note that under zero-latency scenario, there exists at most one honest block at the same level, therefore the total number of adversary blocks across two chains is greater than the number of honest blocks.

Since you can reuse the same leadership certificate across the two chains, but cannot reuse it inside the same chain due to validity condition, the number of adversary blocks in a chain is less than the total number of adversary slots (denote a leadership certificate as a slot). Therefore, total number of adversary blocks across two chains is less than twice the number of total adversary slots, which leads us to the following inequality:

$$\begin{aligned} 2\ no.\ adv.\ slots \geq no.\ adv.\ blocks \geq no.\ honest\ blocks = no.\ honest\ slots \\ \implies 2\ no.\ adv.\ slots \geq no.\ honest\ slots \\ \implies 2\ adv.stake \geq honest\ stake \end{aligned} \tag{14.3}$$

Recall that we use β to represent the fraction of adversary mining power in PoW. Similarly, we can use β in PoS to represent the fraction of adversary stake. Therefore, we have the condition for an attack:

$$\begin{aligned} 2\beta \geq 1 - \beta \\ \implies \beta \geq \frac{1}{3} \end{aligned} \tag{14.4}$$

Now, we can be sure that when β is greater than $\frac{1}{2}$, the system will not be secure, while when β is less than $\frac{1}{3}$, the system will be secure:

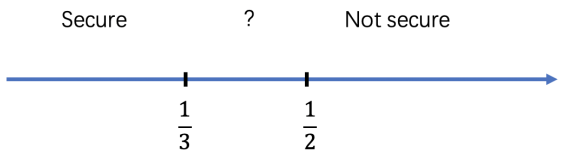

Figure 14.5: Fraction of Adv. on System Security

Right now, we need to deal with the gap between $\frac{1}{3}$ and $\frac{1}{2}$, and there are two ways to close this gap:

1. Show attack when $\beta \geq \frac{1}{3}$
2. Show security when $\beta \leq \frac{1}{2}$, which is the strategy of Ourobros protocol using methods called "Forkable Strings"

14.4 Nakamoto Block Convergence Method

Next, we'll introduce an alternative method for closing this gap mentioned previously, which is also called "Everything is a race" method.

14.4.1 Blocktree Partitioning

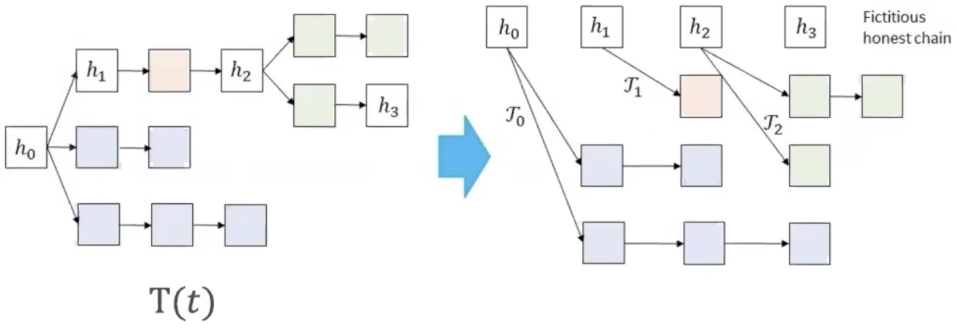

Figure 14.6: Blocktree Partitioning

The first key idea is called "Blocktree Partitioning". In general, we have a block three $T(t)$ which is comprised of many different honest and adversarial blocks (All colored blocks are adversarial blocks) For every block in the tree, you search for the most recent honest block that it is a descendent of. Then we can perform blocktree partitioning on $T(t)$ just like the right part of Figure 14.6 shows.

Generally, we are just viewing the blocktree in a different way. We break off the honest blocks, line up all of them on a single chain, which is called "fictitious honest chain". For adversarisal blocks, we retained how they are connected to each honest block and organize them as a tree with the honest block as the root.

14.4.2 Nakamoto Block

Based on the blocktree partitioning idea introduced above, we can then introduce the second key idea: the Nakamoto block. Recall that in blocktree partitioning, adversarial trees are growing on top of each honest block while the fictitious chain is also growing over time. We first define several notations:

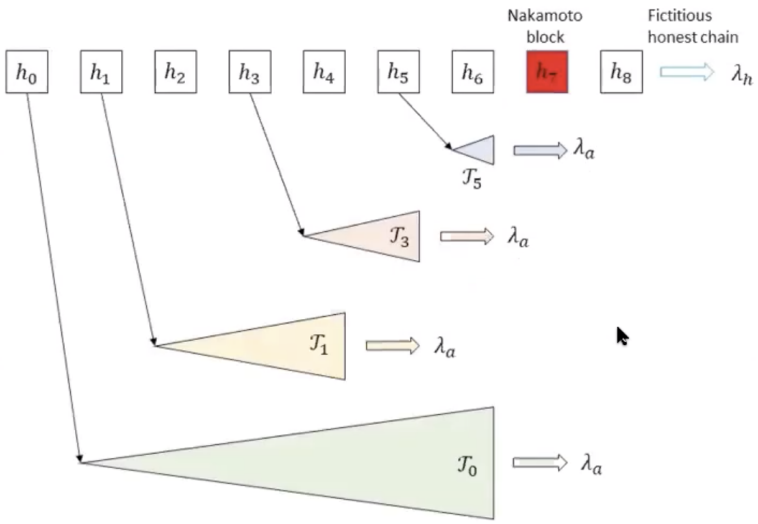

Figure 14.7: Nakamoto Block

1. τ_i: Birth time of honest block i
2. $D_i(t)$: Depth of the tree rooted on honest block i at time t
3. $A_h(t)$: Length of the fictitious honest chain at time t

Then, we have the following definition:

Def.: In fictitious honest chain, a block i can never catch up with block j iff.

$$i + D_i(t) < A_h(t) \qquad \forall t \geq \tau_j \tag{14.1}$$

This means that after τ_j time, honest block growth always wins the race between its competition with the adversary block growth.

On basis of the definition, we can then define the concept of Nakamoto blocks:

Def.: H_j is a Nakamoto block iff. none of the previous blocks can ever catch up.

From the definition of Nakamoto block, we can derive a theorem:

Theorem:

$$If\ H_j\ is\ a\ Nakamoto\ block \Rightarrow H_j\ remains\ in\ longest\ chain\ \forall t \geq \tau_j \tag{14.2}$$

This theorem will be proven in the next lecture, and we can come to a conclusion that we can ensure system security property if there exists a Nakamoto block every now and then in the chain.