ECE 595: Foundations of Blockchain Systems **October 19, 2020**

Lecture 7

Lecturer: Sreeram Kannan *Scribe: Huwan Peng*

Outline: We have shown that the longest chain protocol is secure to private attack under certain conditions and it is also robust to network delay. This lecture is about going beyond private attack. It starts with discussing the security threshold in the longest chain protocol. We then show that increasing security will reduce the throughput of the entire network. Motivated by the trade-off between security and throughput, we introduce a new protocol without this trade-off, GHOST. Though GHOST is secure to private attack and don't have the throughput issue, it is not secure against the balance attack, which is introduced at the end of this lecture.

7.1 Security Threshold

In the last lecture, we analyzed security of the consensus layer with regard to the private attack. When network delay $\Delta = 0$, the probability that a private attack is successful is proportional to e^{-ck} under $\lambda_h > \lambda_a$, where k is confirmation depth. For a general delay Δ, the probability is also proportional to e^{-ck} when $\frac{\lambda_h}{1+\Delta\lambda_h} > \lambda_a$.

7.1.1 Definition

Let us define a new parameter $\beta := \frac{\lambda_a}{\lambda_a+\lambda_h}$, the fraction of adversary's mining power to the total network power. We then use β^* to denote the maximal adversary ratio for security. We call it the *security threshold*.
When network delay $\Delta = 0$, $\beta^* = \frac{1}{2}$ since $\lambda_a < \lambda_h$. When $\Delta \neq 0$, we need $\frac{\lambda_h}{1+\lambda_h\Delta} > \lambda_a$ for the security. At equality, we have $\lambda_h = \lambda_a(1 + \lambda_h\Delta)$, so

$$\beta^* = \frac{\lambda_a}{\lambda_a + \lambda_h} = \frac{1}{2 + \lambda_h\Delta}$$

where λ_h is the honest node block generation rate with a unit of blocks/sec, $\lambda_h\Delta$ means the number of blocks generated by honest nodes in network latency.

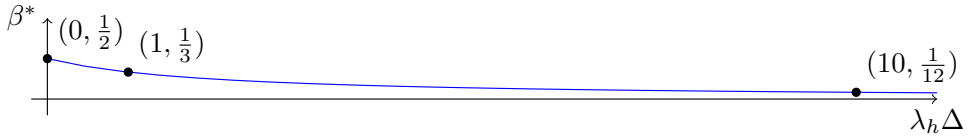

Figure 7.1: Security threshold (β^*) v.s. Number of simultaneous honest blocks ($\lambda_h\Delta$)

Figure 7.1 shows how the security threshold changes with the number of simultaneous honest blocks. For Bitcoin, the honest mining rate λ_h is 1 block per 10 minutes, and the delay Δ is around 10 seconds. Thus, we have $\lambda_h\Delta = \frac{1}{60}$, and the security threshold $\beta^* = \frac{1}{2+\frac{1}{60}} \approx 0.49$.

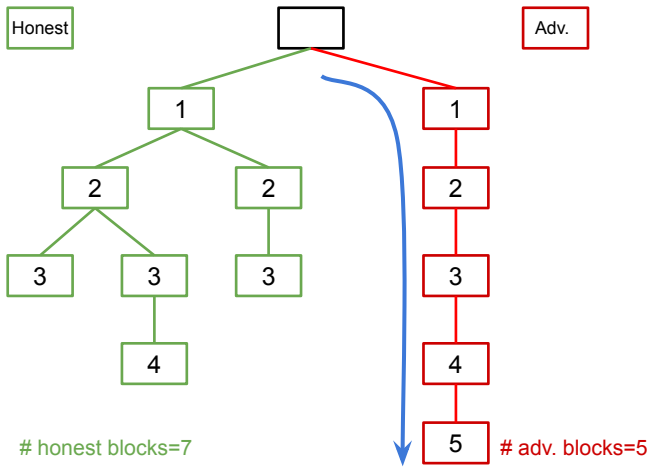

Figure 7.2: A private attack example in the longest chain protocol. The number in each block represent the height of that block. The blue curve indicates the main chain.

7.1.2 Why does the security threshold take a hit?

Assume the honest mining rate is λ_h and the adversary mining rate is λ_a. The total number of blocks produced by the honest nodes in time T is $\lambda_h T$, and the adversary has $\lambda_a T$ blocks. The problem is the adversary could produce blocks sequentially while the honest nodes have forking issue. Figure 7.2 shows an example in which there are 7 honest blocks and only 5 adversarial blocks. However, since the adversarial chain has the height of 5 and honest chain only has the height of 4. The private attack succeed.
So this is what happens when we increase λ_h, $\lambda_h \Delta$ becomes larger, and we get the forking issue. We can control both λ_h and λ_a by changing mining hash threshold. With a larger hash threshold, the mining becomes easier, both λ_h and λ_a increase proportionally. However, the security threshold β^* decreases, the system becomes less secure.

7.2 GHOST Protocol

7.2.1 Motivation

First, let's understand the throughput and latency of the Nakamoto longest chain protocol. For a simple model, we assume each block contains T transactions. The throughput (R) is defined as transaction per second, can be computed by

$$\begin{aligned} R &= Rate\ of\ blockchain\ growth \times T \\ &= \frac{\lambda_h}{1 + \lambda_h \Delta} \times T \end{aligned}$$

where the unit of rate of blockchain growth is blocks per second.
From the above equation, we find that better throughput can be obtained by increasing λ_h or T. However, if we keep increasing λ_h, the denominator will reach saturation. And if we keep increasing T, it will take too long to transmit a block. This is because the delay Δ includes a both a fixed component and a varying component in reality. It can be expressed as

$$\Delta = D + \frac{T}{C}$$

D is the fixed propagation delay and $\frac{T}{C}$ is the varying download delay, where C denotes the download speed (transactions per second). Substituting this into the throughput equation, we have

$$R = \frac{\lambda_h T}{1 + \lambda_h (D + \frac{T}{C})}$$

If T is too large, the download delay will dominate the total latency

$$\lim_{T \to \infty} R = \lim_{T \to \infty} \frac{\lambda_h T}{1 + \lambda_h (D + \frac{T}{C})} = C$$

We will only be able to reach the download speed C, which depends on the network bandwidth. This limitation to throughout seems to be fine, while a more serious problem brought by a large T is the security threshold β^*

$$\lim_{T \to \infty} \beta^* = \lim_{T \to \infty} \frac{1}{2 + \lambda_h (D + \frac{T}{C})} = 0$$

The above equation shows that if we want to reach the throughput capacity, there will be no security in the network. It will not be secure to any adversaries. So there is a trade-off between throughput and security.

$$throughput \uparrow \Rightarrow security \downarrow$$

It is impossible to tune parameters in the longest chain protocol to achieve both high throughput and security. The next question is can we create a protocol without this trade-off?

7.2.2 The heaviest subtree

One of the earliest effort to for this problem is GHOST [1] by Sompolinsky and Zohar in 2015. GHOST is a new blockchain protocol stands for *Greedy Heaviest-Observed SubTree*, which introduces a new policy for the selection of the main chain in the block tree.

As we can tell from Figure 7.2, the honest chain is overtaken when adversarial chain is released because the adversarial chain is longer. This is what would happen in the longest chain protocol, even though the adversary didn't make half of the blocks. Based on this observation, GHOST suggests that blocks that are off the main chain can still contribute to its weight. Instead of using the length to determine what's the main chain, GHOST uses the entire weight of a subtree.

Figure 7.3 shows the same private attack example as Figure 7.2 in the GHOST protocol. When the chain starts at genesis, it has two subtrees with weight of 7 and 5 respectively. We will choose the left (honest) one since it has a higher weight. This selection will continue whenever there are new blocks to be added to the main chain and we will always pick the one with highest weight. This is why it called the greedy heaviest observed subtree.

In GHOST, the private attack succeeds if and only if the number of adversary blocks is greater or equal to the number of honest blocks. Roughly saying, this means adversary power is greater or equal to honest power, $\lambda_a \geq \lambda_h$. Note that $\lambda_h \Delta$ or forking dose not affect this result. We can choose $\lambda_h \Delta$ as high as we want and it will not be limited by network delay or anything. Because of these benefits, this protocol has been widely adopted since it came out in 2015. It is part of one of the most famous blockchains Ethereum.

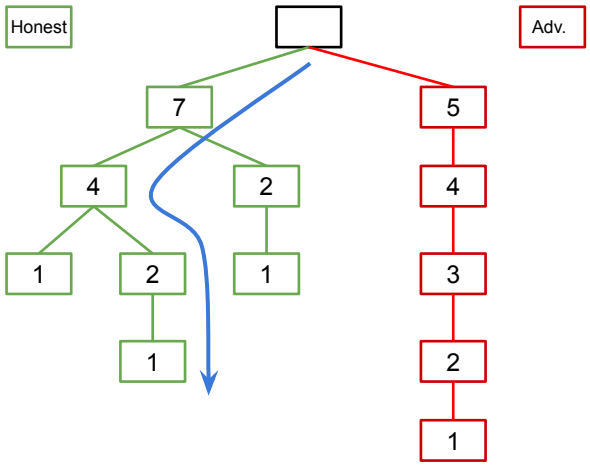

Figure 7.3: A private attack example in the GHOST protocol. The number in each block represent the weight, i.e., the total number of descendants including itself, of that block. The blue curve indicates the main chain.

7.3 Balance Attack

The GHOST protocol is secure to the private attack, as we discussed above. However, it is not secure against other attacks. The most severe of these known attack is called the *balance attack*.
The private attack lets honest nodes confidently confirm a block, then show a competing longer chain. The balance attack is going to be different. It aims to keep the honest nodes perpetually confused between two chains. Once this attack is launched, the honest nodes can never confirm a new block.
We show a balance attack example in Figure 7.4. Assume the total N honest nodes is divided into 2 groups, each with the same mining rate $0.5\lambda_h = 1$. For simplicity, let us consider time running in discrete slots r. Starting from the genesis block, let's say these 2 group of nodes are mining on left and right side of the genesis block respectively. When $r = 1$, they generated block B_1 and B_2 at the same time. At this moment, the left and right sides have the same weight. Assume the two groups generate one block respectively at each time slot until $r = 4$, where only the right side has a new block. This makes the weight of B_2 to 4, while the weight of B_1 is still 3. After this event, all honest nodes from two groups would immediately switch to B_2, then they will only mine on the right chain. In summary, if we just have honest nodes, the main chain will converge on the longest chain even though at some points that may be multiple chains. When there's only one block generated at a time step due to randomness , everybody will confirm that's the longest chain. So the GHOST works fine when there are all honest nodes.
Next let's add adversary nodes to this system, as shown in the right part of Figure 7.4. The goal for adversary is to amplify the confusion already prevalent. To achieve its goal, adversary has been silently mining blocks right on B_1 and B_2, as shown in the red blocks since $r = 1$. All these adversarial blocks are private, which means they haven't been connected to B_1 or B_2. We know at time step $r = 4$, only the right chain gets a new blocks. When the adversary see this, it makes the adversarial block on B_1 public. Note at this moment the adversarial block on the right side remains private. After that, both the weight of B_1 and B_2 become 4, honest nodes still cannot decide the main chain. By balancing the weight of B_1 and B_2, the balance attack makes honest nodes never able to confirm new blocks.

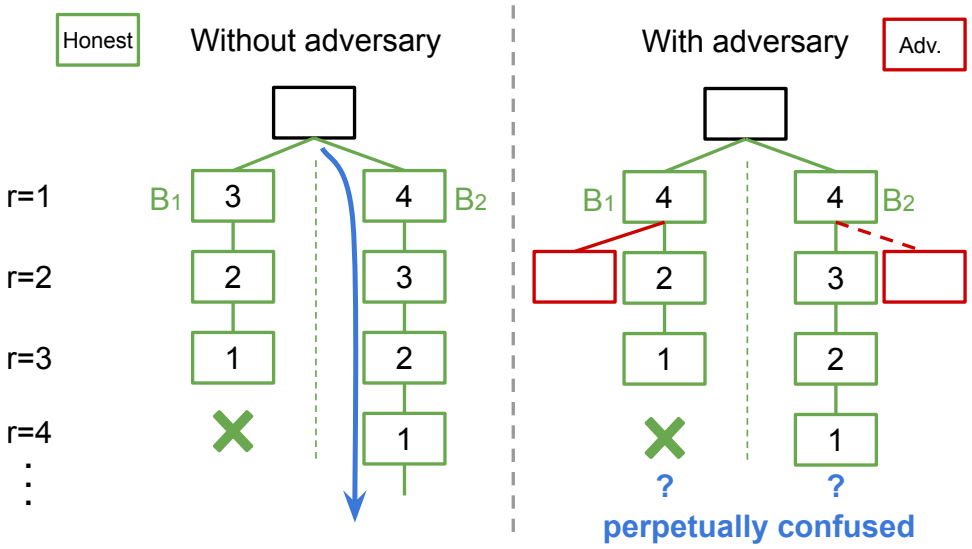

Figure 7.4: A balance attack example in the GHOST protocol. Without adversary, the GHOST protocol works just fine. With adversary, it will mine on both sides on B_1 and B_2 and keep the honest nodes perpetually confused between the two chains.

Let the random variable $H_1[r]$ denoting the number of blocks minded in time step r by the honest node group 1, which are the descent of B_2. And $H_2[r]$ denoting the number of blocks minded by the honest node group 2. Essentially, what the balance attack does is to mine blocks to match the difference $|H_1[r] - H_2[r]|$. If

$$\mathbb{E}\{|H_1[r] - H_2[r]|\} < \lambda_a \Delta$$

where $\lambda_a \Delta$ is the expected number of blocks made by adversary in one time slot, then it is likely adversary has enough blocks to match the difference and keep balance. As a result, the mining power needed by adversary could be way less than 50% of the total power.
We have introduced the security of a blockchain is composed of two things: **Safety**, which means confirmed things remain confirmed; and **Liveness**, which means new honest transactions will be eventually confirmed. The private attack is safety attack and the balance attack is a liveness attack. We will dive much deeper into more general attacks in the next lecture.

References

[1] Yonatan Sompolinsky and Aviv Zohar. Secure High-Rate Transaction Processing in Bitcoin. In *Financial Cryptography and Data Security*, volume 8975, pages 507–527. 2015.