EE374 Internet-scale Consensus **Stanford, Winter 2021**

Homework #3*

Due: Fri, 12-February-2021, 11:59pm – Gradescope entry code: `N8XV23`

Please upload your answers timely to Gradescope. Start a new page for every problem. Latexing your answers is not necessary; handwritten is fine. For the programming/simulation questions you can use any reasonable programming language (please no assembly, brainfuck, etc. ☺). **Comment your source code and include the code and a brief overall explanation with your answers.** A tentative point distribution (in % of the total) is provided in brackets. For most problems there is more than one valid way of solving them!

1. (30%) The chain quality of a blockchain is defined as the fraction of blocks in the public longest chain that are mined by honest nodes. Suppose that the honest mining rate is λ_h and the adversarial mining rate is λ_a. Assume zero delay ($\Delta = 0$) so that all honest blocks will immediately be known to everyone. In lecture we have seen the following lower bound on the chain quality:

$$\mathsf{CQ} \geq 1 - \frac{\lambda_a}{\lambda_h} = \frac{1-2\beta}{1-\beta}, \tag{1}$$

where $\beta = \frac{\lambda_a}{\lambda_a+\lambda_h}$.

   a) Recall the *selfish mining* strategy: The adversary always mines on the longest chain, but keeps their blocks private. Every time an honest block is mined, the adversary displaces the block by releasing one of their private blocks (if it has meanwhile pre-mined at least one adversarial block). Assume that when there are multiple longest chains of equal length, the adversary can choose which longest chain the honest miners will adopt. Under this assumption, the most favorable situation is that the adversary's block becomes the new longest chain where honest miners will mine their next block. If the adversary does not have any blocks in private when an honest block is mined, it starts a new private chain on the new honest block.

      i. Suppose honest (h) and adversarial blocks (a) are mined in the order:

$$a_1, h_1, a_2, a_3, h_2, h_3, h_4, a_4, h_5.$$

      Draw the evolution of the block tree under this attack strategy, always including both honest and adversary blocks. Indicate the longest chain after all these blocks have been mined.

*Version: 1 – Last update: 5-Feb-2021

ii. Simulate the system and plot the CQ as a function of β to confirm that the above strategy achieves the chain quality in the lower bound, as argued in class.

b) Now assume that *when there are multiple longest chains of equal length, ties are broken randomly.* In particular, the adversary's strategy is as follows. The adversary mines a chain in private. Every time an honest block is mined, the adversary releases one of their private blocks (if they have one). With probability 1/2 all honest miners continue to mine on the tip of the honest chain, and with probability 1/2 all honest miners mine on the adversary's newly released block. In either case, the adversary continues to extend its own chain, in private. If the adversary does not have any blocks in private when an honest block arrives, then the adversary starts a new private chain on the latest honest block.

i. Suppose honest (h) and adversarial blocks (a) are mined in the order:

$$a_1, h_1, a_2, a_3, h_2, h_3, h_4, a_4, h_5.$$

Also suppose that the honest block is chosen as the longest chain when h_1 appears, and the adversary's block is chosen when h_2, h_3, h_5 appear. Draw the evolution of the block tree under this attack strategy, always including both honest and adversary blocks. Indicate the longest chain after all these blocks have been mined.

ii. Simulate this strategy and plot the resulting chain quality as a function of β. Compare this with the chain quality from part (a), and the ideal chain quality if the protocol were completely fair, in the same plot. Comment on your results.

2. (20%) In the safety argument for Streamlet, we considered a scenario that 3 blocks b_5, b_6, b_7 in epochs $5, 6, 7$ respectively are notarized and hence blocks b_5 and b_6 are confirmed. We argue that if the number of adversary nodes $f < n/3$, then there cannot be another block from epoch $4, 5, 6$ or 7 notarized at the same level as b_6.

   a) Argue that there cannot be any block from any epoch less than 4 notarized at the same level as b_6.

   b) Argue that there cannot be any block from any epoch greater than 7 notarized at the same level as b_6.

3. (20%) In Streamlet, the quorum size for notarization is chosen to be $2n/3$, where n is the number of (permissioned) nodes. With that quorum size, we showed that the protocol can tolerate up to $n/3$ adversary nodes. In this question, we explore whether the quorum size can be optimized to increase the resilience of the protocol, i.e., the number of adversary nodes it can tolerate.

   a) Suppose we set the quorum size to be q. Let f be the number of adversary nodes. What condition must q and f satisfy for Streamlet to be safe? Explain.

   b) What condition must q and f satisfy for Streamlet to be live? Explain.

   c) Using the constraints in parts (a) and (b), optimize q to maximize the resilience of Streamlet. What is the resulting resilience?

4. (30%) Consider the basic version of Prism we discussed in Lecture 7, where there are two types of blocks, transaction blocks and proposer blocks. The proposer blocks contain hashes of the transaction blocks, and the leader proposer blocks are selected simply by the longest chain rule among the proposer blocks themselves, i.e., the proposer blocks on the longest chain in the proposer tree form the ordered sequence of hashes of the transaction blocks.

   a) Suppose the communication bandwidth per node is 20 Mbits/second and each transaction block is 20 KBytes in size (like the blocks in Ethereum). On the average, how many transaction blocks is each node downloading each second if the system is operating at the communication physical limit? You can ignore the communication requirement of proposer blocks for now.

   b) If one transaction is 500 Bytes in size, what is the total throughput of the system in terms of transactions per second? (Ignore block headers here.) How does this compare to Ethereum, where the block mining rate is 1 block every 15 seconds?

   c) If the proposer blocks in Prism are mined at 1 block per 15 seconds, what is the average number of transaction blocks each proposer block refers to? You can assume there is no forking in the proposer tree. If one thinks of the combination of the proposer block (as a header) with the transaction blocks it refers to (as the content) as a superblock, how big is this superblock?

   d) Now suppose a fraction of adversarial nodes with hashing power β fraction of the total hashing power is trying to do selfish mining on the proposer chain. According to our result in the class, the adversary is able to lower the chain quality of the honest nodes to $(1-2\beta)/(1-\beta)$ on the proposer chain. Moreover, the adversary's proposer blocks only refer to its own transaction blocks. (It's selfish, after all.)

      i. What fraction of the total *transaction blocks* are mined by the adversary?

      ii. On the average, *at most* how many transaction blocks does each adversarial proposer block point to? How big are these adversarial superblocks?

      iii. On the average, *at least* how many transaction blocks does each honest proposer block point to? How big are these honest superblocks?

      iv. Is Prism fair in terms of the transaction throughputs of each party, despite the unfair chain quality? Discuss any differences with Bitcoin.