COMS 6998-006 (Foundations of Blockchains): Homework #5

Due by 11:59 PM on Thursday, November 4, 2021

Instructions:

(1) Solutions are to be completed and submitted in pairs.

(2) We are using Gradescope for homework submissions. See the course home page for instructions, the late day policy, and the School of Engineering honor code.

(3) Please type your solutions if possible and we encourage you to use the LaTeX template provided on the Courseworks page.

(4) Write convincingly but not excessively. (We reserve the right to deduct points for egregiously bad or excessive writing.)

(5) Except where otherwise noted, you may refer to your lecture notes and the specific supplementary readings listed on the course Web page *only*.

(6) You are not permitted to look up solutions to these problems on the Web. You should cite any outside sources that you used. All words should be your own. Submissions that violate these guidelines will (at best) be given zero credit, and may be treated as honor code violations.

(7) You can discuss the problems verbally at a high level with other pairs. And of course, you are encouraged to contact the course staff (via the discussion forum or office hours) for additional help.

(8) If you discuss solution approaches with anyone outside of your pair, you must list their names on the front page of your write-up.

(9) Some of these problems are difficult, so your group may not solve them all to completion. In this case, you can write up what you've got (subject to (4), above): partial proofs, lemmas, high-level ideas, counterexamples, and so on.

Problem 1

(20 points) In Lecture 14 we described Bitcoin as a longest-chain protocol, with honest participants working to extend the longest chain.

(a) (8 points) You might think that the "longest" chain is defined as the chain with the largest number of blocks. In fact, Bitcoin uses a more subtle definition. How exactly is the longest chain determined in Bitcoin? Be specific — e.g., it should be clear from your description whether or not a light client (knowing only the block headers) is capable of figuring out which chain is the longest. (Cite the source(s) you used to answer this question; just the URLs are fine.)

(b) (8 points) Suppose Bitcoin used the more obvious definition of a longest chain (defined solely by the number of blocks). Propose how a miner with a relatively small amount of hashrate could attack this more naive implementation.

(c) (4 points) Suppose there is a tie between two or more chains for being the longest (according to the definition in (a)). In Bitcoin, how are honest miners supposed to decide which one to extend?

[Hint: check the Bitcoin white paper.]

Problem 2

(15 points) This problem considers an alternative to the longest-chain rule for resolving forks. For this problem, for simplicity, assume that the difficulty target is the same for a blockchain's entire existence. Given an in-tree G (a tree with all edges directed toward the root, the genesis block), the longest-chain rule singles out the blocks on the longest leaf-root path as those that are (at least tentatively) confirmed.

An alternative approach, known as the GHOST rule, is to define the chain of valid blocks inductively, starting from the root and ending at a leaf. The genesis block is always included. Inductively, suppose that v was just included in the chain-so-far, and that the vertices $w_1, \ldots, w_k$ point to it. Then exactly one of the w_i's is included—the block with the largest number of descendants (i.e., the largest induced subtree). (Don't worry about how ties are broken.)

(a) (5 points) Give an example of an in-tree for which the longest-chain rule and the GHOST selection rule single out different chains of blocks. (Construct your example so that there are no ties with respect to either rule.)

(b) (5 points) Suppose we modify Bitcoin (with the usual longest-chain rule) so that it targets an average block rate of one per second rather than one per 10 minutes. Explain why an attacker with only 49% of the hashrate would then typically be able to successfully execute the canonical double-spend attack (as the attack was described in Lecture 13, or in the Bitcoin white paper).

(c) (5 points) Why might the GHOST rule from part (a) help mitigate the issue identified in part (b)?

Problem 3

(45 points) This problem concerns the sampling of a block producer in a proof-of-stake blockchain. (For concreteness, let's think about longest-chain such blockchains.) For this problem, assume that a perfectly uniformly random number $r \in [0, 1]$ falls from the sky, and the goal is to sample one of n stakers that have stake values $s_1, s_2, \ldots, s_n$. (So the ith staker should be sampled with probability $s_i / \sum_{j=1}^{n} s_j$.)

(a) (4 points) In the "follow-the-satoshi algorithm," the number r is used to sample one coin c (in the smallest-possible denomination, e.g. a satoshi in Bitcoin or a wei in Ethereum) uniformly at random. Here by a "coin" we mean a specific coin among those created in a specific coinbase transaction (the transaction that mints new money for the block reward). (Don't worry about the details of how to go from r to c, just assume that you have c). Then:

(i) find the current owner pk of the coin c;

(ii) if pk is one of the stakers and c is one of the coins that they staked, make pk the next block producer;

(iii) if not, rechoose a random coin and try again.[1]

Briefly prove that the follow-the-satoshi algorithm would sample a staker with the desired distribution.

(b) (12 points) Consider a UTXO-based blockchain like Bitcoin (see Lecture 14). Discuss in detail how you might implement step (i) of the follow-the-satoshi algorithm and the potential challenges involved.

[Things to discuss: is (i) even well defined? If not, is there some way to make it well defined? Conceptually, starting from a coinbase transaction, how do you find the current owner of that coin? What data structures seem most appropriate for implementing (i) efficiently?]

(c) (5 points) Here's another way to implement stake-based sampling (assuming access to perfect randomness) in a proof-of-stake blockchain:

– For each staker $i = 1, 2, \ldots, n$: (in arbitrary order)

[1] Assume that the random string r can be reused to produce any number of independent and uniformly random coins (sampled with replacement). Typical proof-of-stake blockchains have at least several per cent of the currency staked (and often as high as 70%), so a few dozen tries is generally enough to sample a coin owned by a current staker.

* Flip a biased coin with probability

$$\frac{s_i}{\sum_{j=i}^{n} s_j}$$

of "heads."

* If the coin comes up "heads," deem user i the next block producer and halt.
* Otherwise, continue.

Prove that this algorithm selects a staker as the next block producer with probability proportional to their stake.

(d) (10 points) Propose a data structure that you could use to keep track of account balances and implement the sampling in (a). Your data structure should support account balance updates and sampling from the distribution in (a) in $O(\log n)$ time, where n is the number of different accounts.

[Hint: for the sampling operation, you might want to develop an alternative view of the procedure in (c) that uses a single uniform random number from $[0, 1]$, rather than a sequence of n biased coin flips.]

(e) (6 points) Suppose you instead wanted to implement a hash-based lottery, with a staker i selected for block production if and only if SHA-256$(sign_i(pk_i) \,||\, t \,||\, r_t)$ is at most some threshold τ_i. (Here pk_i denotes i's public key, $sign_i$ indicates a signature with i's secret key, t is the time step, and r_t is a random seed, which you should assume is perfectly random (from $\{0,1\}^{256}$, say).)

Suppose you wanted, in expectation, one block producer per time step. How should you set τ_i, as a function of $s_1, s_2, \ldots, s_n$?

(f) (8 points) Is the result of the random selection procedure in (e) identical in all respects to those in (a) and (c)? If not, propose some advantages and disadvantages of the sampling method in (e) relative to those in (a) and (c).

Problem 4

(15 points) Recall that in Lecture 13, our analysis suggested that a 51%-type attack would be profitable if and only if

$$V > \underbrace{\beta RT(f(\beta) + \Delta)}_{\text{economic security}},$$

where V is the off-chain value of the attack (e.g., from a double-spend), R is the average per-block reward in USD terms, T is the duration of the attack under consideration, $f(\beta)$ quantifies the extent to which there are increasing costs of acquiring more hashrate or coins, and Δ is the drop in coin price after a successful attack. (Below, as always, cite the sources you used for your analysis.)

(a) (5 points) Compute the current economic security (i.e., the right-hand side above) of Ethereum with respect to one-hour-long 51% attacks. Assume that $f(\beta) + \Delta = .2$.

(b) (5 points) To what extent is the value of R under the control of a protocol designer? Give answers for two different regimes: one in which transaction fees are always low (zero, say), and one in which they are potentially large.

(c) (5 points) The Ethereum Classic blockchain has been 51%-attacked several times. Pick three of these attacks, and compute what Δ was for each of them. (I.e., looking at historical price data, examine the coin price just before each attack and the coin price shortly thereafter (e.g., 24 hours later)). Is Δ higher or lower than you expected?