Problem Set 2: The Blockchain

Cyrus Malekpour (cm7bv@virginia.edu)

Problem 1

Satoshi makes several assumptions in this section. Firstly, he assumes that the coinbase transactions are still yielding bitcoin, as these are a large portion of the current mining rewards. Transactions fees are currently only small compared to the coinbase transactions. Satoshi also assumes that mining is profitable, which may not be the case for the attacker as the difficulty increases. It is possible that the attacker reached 50%+ hashing power while operating at a loss, making it infeasible for them to continue their hashing while operating honestly.

Problem 2

This refers to the number of blocks that an honest transaction receipient should wait until they can be sure (with $p < 0.001$) the attacker cannot reverse it to pay themself. In this case, these numbers mean that if the attacker has probability 0.45 of finding the next block, an honest receipient should wait for 340 blocks to ensure that the probability of an attacker being able to reverse the transaction is < 0.001

Problem 3

```
P < 0.05
q=0.10  z=3
q=0.15  z=3
q=0.20  z=5
q=0.25  z=6
q=0.30  z=10
q=0.35  z=17
q=0.40  z=36
q=0.45  z=137
```

Problem 4

a. There is no way of verifying the order of the data. Assuming the data itself does not contain *i*, the index, there is no ordering of data built into what is cryptographically verified.

b. We can only verify if the data matches the signature and if we signed it. Technically, we cannot be sure it is the data we wrote; we can only be sure it was some information that we signed at some point. It is possible that all data elements could be randomly shuffled, and we would not be able to tell. However, if the signature does not match the data we can be sure it is *not* what we wrote.

Problem 5

a. To write, you pull all data from the remote server, concatenate your new data to it, hash all of it, and then save the new data to the remote server. Reading involves pulling all data from the server, so you can verify it, and then reading the section you want to. To verify, you have to pull all data from the remote server, concatenate and hash all of it, then compare to your local hash.

b. The cost of reading and writing scales linearly as *n* increases. Reading and writing both require pulling all existing data and hashing it, and the number of records to pull increases linearly with *n*.

Problem 6

a. To write, hash the new record and add it as a leaf to the merkle tree. Assuming you store the hashes of the tree locally, you can do this in log(n) update operations, where n is the total number of records in the tree.

Reading from the tree can again be done by performing log(n) operations, corresponding to the number of hash operations needed to verify the record you read.

Verifying the entire tree involves pulling all records, arranging them appropriately in a tree structure, and hashing the left and right branches together up the tree. Finally, the computed root tree hash can be compared to the same hash to verify all the data.

b. Reading and writing both scale logarithmically with the number of records. Both require log(n) operations to either update the local merkle tree or to verify a read.

Problem 7

a. If a node has 15% of the hashing power, it should be expected to find 15% of all total blocks per day.

Assuming a block is found every 10 minutes, and the difficulty does not change, there should be (24 hour/day * 60 minute/hour * 0.1 block/minute) = 144 blocks per day. 15% of 144 is 21.6, so roughly **21** blocks per day.

b. A mining pool with *α* percent hashing power should find **(α * t / 10)** blocks in *t* minutes

Problem 8

Assume we are an honest mining pool with hashing power *α*, and the latency between us and the other supernode is *L*. Given that 144 blocks are found per day, we can assume the other mining pool will find *(1 - α) * 144* blocks per day. For each of those blocks, we have *L* seconds to find another block to create an orphan one. For each of these periods, we have *(α * L / 600)* chance to find a block. Given that there are 144 of these periods per day, we should expect to find **144 * (α * L / 600)** orphan blocks per day, from the perspective of our supernode.

Problem 9

If a miner is selfish mining, with $\alpha < 0.5$ of the network hash power, orphaned blocks will not appear at somewhat consistent intervals, but rather will appear in chunks as the selfish miner reveals its entire lead to stop the honest supernode from catching up. The overall number of orphan blocks will increase. This is because if the selfish mining supernode finds a block, it will not broadcast it immediately. If the honest node finds a block before the selfish supernode finds another, there is a guaranteed orphan block. If the selfish node continues to find *n* blocks before its lead again drops to 1, there will be *n-1* orphan blocks. We are guaranteed at least 1 orphan block whenever the selfish supernode finds a block, and $\alpha * 144 > 144 * (\alpha * L / 600)$ if $L < 600$. In other words, if the latency *L* is less than 10 minutes then we should find more orphan blocks in the network when one supernode is selfish mining.

Problem 10

I chose to do Problem 10b, and my question is intended for the law professor.

"We have seen that there are several methods for encoding data into the blockchain, the first of which was a newspaper headline by Satoshi. Since the blockchain in its entirety is stored by all full nodes, what would be the legal ramifications of someone encoding child pornography or another illegal-to-possess piece of data?"