Sugat Poudel
CS 4501: Crypto Currency Cabal

Problem Set 1: The Blockchain

Problem 1
What are the assumptions necessary to support Satoshi's claim that it is more profitable for a greedy attacker with a majority of the mining power "to play by the rules"? (other than the assumption that the greedy attacker is a "he")

Satoshi first makes the assumption that the sum transactions of the attacker will on average be less than the reward for finding the block thus it would be financial sense to mine the block instead of reverting their old transactions. There is an underlying assumption that the sum of transaction fees will always be enough financial incentive even after the mining fee goes away after factoring for inflation. Moreover, there is a firm assumption that the incentive to mine honestly in the blockchain rather than a rogue chain will always be financial rather than a more personal or even social reason.

Problem 2
At the end of Section 11, Satoshi presents a table for $p < 0.001$, where it is listed "q=0.45 z=340". What does this mean in plain English, expressed in a short sentence?

If the attacker has 45% of the network hashing power, then it will take 340 block lead for the honest block chain to ensure that the probability that the attacker catches up is less than 0.01%.

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

For this problem, I implemented Satoshi's AttackerSuccessProbability function to find the probability that the attacker could catch up in python and then constructed a function that ran the original function with varying values of z until the probability fell below the specified threshold. My function was as follows:

```
def attacker_low_success(q, P):
  z = -1
  prob = 1.0
  while not prob < P:
    z += 1
    prob = attacker_success_probability(q, z)
  return z
```

Problem 4

a. There is a problem with this scheme unless the bytes of data[i] indicate that it is in fact from i-th index. What is this problem?
   I assumed that the same private key is used to sign all records in the cloud. Unless your data explicitly houses its index, then any data written can be returned from the database and since everyone is signed with the same private key, they would all be successfully verified. So we don't know if the data being returned is the actual data we wrote at that particular index.

b. Suppose we perform a write at position i, both data and signature. Later on, when we read it back, if the signature matches the data, can we be sure that it is indeed the data item we wrote? Explain.
   If we assume that the stored record is the concatenation of the signature of the data, there will be an intrinsic disconnect between the signature and the data in a way that the cloud service can easily manipulate the content such that even if the signature is verified the corresponding data could still be erroneous. Moreover, if signature is simple a digital key stored in the database, it does not carry any cryptographic value for the data because there is no implementation of encumbrance that is the data isn't actually signed.

Problem 5

a. What is the write/read/verify procedure for this system?
   - Write: direct write of data in the database for given index, then read all other records in the database, concatenate them together, hash the result and store resulting hash locally
   - Read: direct read of data in the database for given index (is read/ verify one)
   - Verify: read all data from the database even if reading only from one index, concatenate them all together, hash the result then verify the hash with the hash stored locally

b. How does the cost of reading and writing to the database scale with n (the number of records)?

   Reading will always occur in constant time as you're just reading data present directly in the database however verification will take linear time as each record will need to be read and the sum concatenation will need to be hashed. Similarly, writing will scale linearly as the number of records, n, increases because with each write you have to deal with every single record in the database once when concatenating and updating resulting hash.

Problem 6

a. What is the write/read/verify procedure for this system?
   - Write: direct write of data in the database for the given index, add the hash of the data as a new leaf in the merkle tree locally and percolate up while updating the hash of each successive parent node in $\log n$ time
   - Read: direct read of data in the database for the given index
   - Verify: given a data record for a particular index, find the hash then get the hash of adjacent node, concatenate the previous two hash and find the hash of the sum, verify that the new hash matches that of the parent node. Repeat this process up till the root node (merkle root). If all resulting hash match the merkle tree than data integrity has been preserved.

b. How does the cost of reading and writing to the database scale with n (the number of records)?

   Reading is still constant operation as a record from the database is read directly for a given index however verification will take linear time because a single path will be traversed and the hash value of each node will be compared against hashing concatenation of adjacent node values. Similarly, write scales logarithmically with n because in order to verify a given data, you only need to traverse one path in the merkle tree, which is of height $\log n$, and interact with adjacent nodes.

Problem 7

a. If a mining pool has 15% (α = 0.15) of the total network hashing power, how many blocks is it expected to find in a day?

   In a given day, we can expect 144 blocks to be mined based on the assumption that a block is mined every 10 minutes. If a pool has 15% of the hashing power, we can assume that it will mine 15% of the blocks mined that day thus the pool can expect to find 21 blocks for a given day.

b. Obtain a general formula for expected number of blocks a mining pool with α fraction of the total hashing power should find in *t* minutes.
   If E is the expected number of blocks a mining pool should find:

$$E = t\,\cancel{\min} \times \frac{1\,block}{10\,\cancel{\min}} \times \alpha = \frac{t * \alpha}{10}\ blocks$$

Problem 8
Assuming all of the miners are honest, what is the expected number of orphaned blocks per day for an honest mining block with hashing power α and latency L (as simplified above).

With the assumption that there is no latency within the node representing the rest of the bitcoin network, any blocks mined there will be automatically accepted into the block chain unless the mining pool mines a valid block first. The rest of the network comes ahead of the mining pool if they mine a block within a latency period for the pool. We can use the expression derived in 7b to determine the expected number of blocks that will be mined within a latency period by the rest of the network.

$$E_{lat\_net} = \frac{L}{600}(1 - \alpha)$$

Then, since a latency period will come each time the pool mines a block, there will be 144α latency periods and thus the expected number of orphaned blocks from the mining pool in all those periods will be.

$$E_{orphan\_pool} = E_{lat\_net} * 144\alpha = 144\alpha * (1 - \alpha) * \frac{L}{600}$$
$$E_{orphan\_pool} = \frac{144L\alpha(1 - \alpha)}{600}$$

We can extend this to find the total number of orphan blocks between the two nodes by finding the orphaned blocks from the rest of the network and adding that to the orphaned blocks from the pool.

$$E_{lat\_pool} = \frac{L\alpha}{600}$$

$$E_{orphan\_net} = E_{lat\_pool} * 144(1 - \alpha) = 144(1 - \alpha) * \alpha * \frac{L}{600}$$

$$E_{orphan\_net} = \frac{144L\alpha(1-\alpha)}{600}$$

$$E_{orphan} = E_{orphan\_net} + E_{orphan\_pool}$$

$$E_{orphan} = \frac{2 * 144L\alpha(1-\alpha)}{600}$$

Problem 9

How does this change if the mining pool is mining selfishly?

If the mining pool is following a selfish strategy, the expected number of orphan blocks would always be one less than the number of blocks mined by the pool and they will all be from the rest of the network. Following the selfish strategy from the slide set in class 11. The expected number of blocks mined for the two nodes would be:

$$E_{pool} = \alpha^2 \times \left(2 + \frac{\alpha}{1-2\alpha}\right)$$

$$E_{net} = 1 - \alpha^2$$

Since we assume that orphan blocks will be one less than the blocks mined by the pool, we can express the number of orphan blocks as a proportion as follows

$$E_{orphan} = \frac{\alpha^2 \times \left(1 + \frac{\alpha}{1-2\alpha}\right)}{1 - \alpha^2 + \alpha^2 \times \left(2 + \frac{\alpha}{1-2\alpha}\right)}$$

Using this proportion, we can multiply it by 144 to find the expected number of orphaned blocks in a day given that the mining pool is mining selfishly.

$$E_{orphan\_day} = \frac{\alpha^2 \times \left(1 + \frac{\alpha}{1-2\alpha}\right)}{(1-\alpha^2) + \left(\alpha^2 \times \left(2 + \frac{\alpha}{1-2\alpha}\right)\right)} \times 144$$

We don’t have to consider latency from the pool to the rest of the network because the selfish pool will not publish blocks unless it has a sufficient lead, in this case 2 blocks.

Problem 10

a. In class, we saw how pool-hopping can be used to game a “proportional” reward scheme. Design a simple reward scheme that eliminates pool-hopping incentive. In particular, derive the expected reward for your scheme and show that it does not depend on time since last block (or any variable controlled by the miner).

   With the proportional reward scheme, a miner contributing shares early after a pool finds a block will on average have a higher payout. This can lead to exploitation of the pools through pool hopping. One scheme to avoid this practice is to have a flat rate reward for each share to avoid making the payout proportional. This leads to a virtually 0 variance income that eliminates all risk for the miner; all of the risk will instead be transferred to the pool manager/owner. This scheme avoids the reward being dependence on time since block but rather purely on a miner’s contributed effort. However, since it is risky for the pool managers since the miners are still being paid even if they didn’t find a block, they are likely to charge a higher fee.

   If R is the constant flat rate value of each share and S is the number of shares a miner submitted since the last block then the expected reward, E, for each miner is as follows

$$E = R * S$$

   This expected payout is not directly dependent on time

b. FBI: Unfortunately, some people are first exposed to bitcoins to unwillingly pay malicious groups that hijack data such as with the case of cryptolocker. In popular culture, bitcoins have a reputation for being an anonymous currency when in fact the fundamental basis for bitcoins lies in the public ledger meaning every transaction is displayed and handled publicly. Is it possible to locate someone using their bitcoin address especially when might use laundering services? Is there a way to blacklist certain transitions if we know that their end game is malicious without losing the property of decentralization?

   Law: How is a BTC exchange handled differently from a bank in terms of compliance regulations aside from the fact that one is completely virtual?