Fangyang Cui

Problem Set 2

1. If the attacker plays by the rules, he is awarded with newly generated Bitcoins along with transaction fees for every block he mines honestly. On the other hand, if the attacker can perform double spend attacks by creating transactions and later invalidating them. There is an assumption needed for the rational behavior of the attacker to become mining honestly. That assumption is that the monetary value of the new blocks created by honest mining is greater than the monetary value of any double spend attacks the attacker can perform before the value of the Bitcoin network plummets.

2. For an attacker who controls 45% of the network hashing power, the probability that the attacker can invalidate a transaction after the transaction has 340 confirmations is less than .1%.

3. For $P < .05$,

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

4. a) There is no way of knowing that when requesting record i, the record returned by the cloud server is actually record i. Instead, the server could send another valid record k. Unless the data has information about which record number it contains, this will be a problem.

b) No, the data could be from record k instead of record i. The signature would still match since record k is a valid record.

5. a) Writing would require concatenating the old records with the new records and hashing the result. The new result is stored locally as the hash to compare against. The new records are sent to the cloud.

To read a record, a request is made to the cloud server for a particular record. However, in order to verify a record, every record must be read from the cloud storage provider. The data received is concatenated and hashed and compared to the hash stored locally.

b) The cost of reading and writing to the database scales linearly with n.

6. a) To write a record, the hash of the new record is added to the merkle tree as a new leaf node of the tree. The merkle tree node can also store a record ID that is associated with a given record. Along the way back up to the root, the merkle hash must be updated to include the new record. The new record can be sent to the cloud server.

To read and verify a record, the record is requested from the server. The record ID of the record is used to locate the position of the corresponding node in the merkle tree. The hash of the record received from the cloud is computed and concatenated with the hash contained in the sibling node of the merkle tree. The concatenated hashes are then hashed and must match the hash stored in the parent node of the node corresponding to the requested record. This process is repeated until the root of the merkle tree is reached. If the hash at the root matches the computed hash, the record is verified.

b) The cost of reading and writing scale logarithmically with the number of records.

7. a) Total blocks per day found by the network on average = 24*60/10 = 144.

The expected blocks found by the pool is 144 * 0.15 = 21 blocks.

b) α *t/10

8. The mining pool experiences an orphaned block when the rest of the network finds a block and the mining pool finds a block before the network's block has propagated to the mining pool.

On average, the network will find (1- α)*144 blocks per day. In the next L seconds, any blocks found by the mining pool will be orphaned. Therefore, the number of orphaned blocks found by the mining pool is the number of blocks found in those L*(1- α)*144 seconds. This number is

α *(L/600)*(1- α)*144.

9. From the perspective of the network, there is an orphan whenever there are multiple competing blocks are announced. This would happen when the selfish miner gets 2 or more blocks ahead of the honest network and ends up revealing the longer chain. The selfish miner gets 2 or more blocks ahead with α^2 probability. Assuming constant difficulty and network hash rate, there are expected to be 144 blocks per day, so the selfish miner should expect α^2 * 144 occurrences of getting ahead of the network by 2 or more blocks. Once ahead by 2 blocks, the selfish miner can expect to mine an additional α/(1-2 α) blocks before the network catches up to only 1 block behind. Once the network reaches 1 block behind, the selfish miner releases 2+ α/(1-2 α) blocks making 1+ α/(1-2 α) blocks found by the network to be invalid. Therefore, the expected orphaned blocks should be approximately (α^2 * 144) * (1+ α/(1-2 α)). This number is a little lower because when the selfish miner reaches 2 blocks ahead, they spend time trying to increasing their lead so the number of chances per day they reach 2 blocks ahead of the network is lower than (α^2 * 144).

10A. One scheme that makes the pool more vulnerable to variance is to pay per share. Whenever a share is submitted, no matter when it happens, the miner receives a fixed amount depending on the share difficulty compared to the difficulty of the Bitcoin network. If a block must have a hash below D to be valid and the pool accepts all shares that hash below S where S > D, then each share will pay 25*S/D bitcoins. The expected reward is based on each miner’s hashing power and is equal to their proportion of the network's hashrate * 25 * 144 assuming constant difficulty and 10 minutes per block.