Reid Bixler
rmb3yz
CS 4501 Problem Set 2
10/8/15

Problem 1.

One of the first assumptions is that this greedy attacker is a single individual. If this greedy attacker were actually a pool of individuals, it may prove to be more profitable to try to steal back payments for all the individuals of the group rather than to try to split up the profit of the mining reward. Another assumption is that the mining reward is high enough to warrant playing honestly. Eventually the block reward will get to a point where, if there were a greedy attacker, it would most definitely prove to be more profitable to not play honestly simply because the reward for mining would not be enough, even if you include the transaction fees. Another assumption is that the greedy attacker is seeking to simply gain money from this. If the greedy attacker didn't want money with all of that processing power (it would be very weird, but it is possible), then they could intentionally try to undermine the system of Bitcoin for the sake of breaking Bitcoin or for some other weird reason. Another assumption would be that the greedy attacker would be getting more money from the new coins than they would from trying to steal back their own payments. It could be entirely possible that the attacker might have spent hundreds of thousands of bitcoins in the past and that it would be more profitable for them to try to go back on those transactions rather than to try to get them back 50 or 25 bitcoins every 10 or so minutes.

Problem 2.

If the probability an honest node finds the next block is .55 and the probability the attacker finds the next block is 0.45 and the honest nodes have a block chain 340 blocks ahead of the attacker, then the attacker will be able to create their own alternate block chain (with bad transactions included) and eventually catch up to the honest node chain with a probability of less than 0.001.

Problem 3.

$P < 0.05$
$q=0.1 \quad z=3$
$q=0.15 \quad z=3$
$q=0.2 \quad z=5$
$q=0.25 \quad z=6$
$q=0.3 \quad z=10$
$q=0.35 \quad z=17$
$q=0.4 \quad z=36$
$q=0.45 \quad z=137$

PYTHON CODE FOR PROBLEM 3

```
def AttackerSuccessProbability(q, z):
    p = 1.0 - q
    lambd = z * (q / p)
    sum = 1.0
    for k in range(0,z):
        poisson = math.exp(-lambd)
        for i in range(1,k+1):
            poisson = poisson * (lambd / i)
        sum = sum - (poisson * (1 - (q / p)**(z - k)))
    return sum

def main():
    q = [0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45]
    ans = [1, 1, 1, 1, 1, 1, 1, 1]
    print("P < 0.05")
    for i in range(len(ans)):
        z = 0
        while ans[i] > 0.05:
            ans[i] = AttackerSuccessProbability(q[i],z)
            z += 1
        print("q=" + str(q[i]) + "   " + "z=" + str(z-1))
    return 0
```

Problem 4

One naive approach would be to just sign each record data with a private key, and verify the signature on reading it.

(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?

The problem is that they could be storing your data in a way other than in the order that you submitted it. If they store it in the same order you submit it, as in it is indexed from 0 to n-1 from the first to last document in sequential order, then you are able to access each data[i] knowing what to be expecting. However, if the cloud storage does not index their data, then when you try to verify the data on reading it, it would be likely that you are returned a different document than you expect and as such come up with an error.

(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.

It would depend on what type of cryptography that we are using and if there was any chance that your own private key was obtained in between writing and accessing. First if you are using a poor encryption for the private key that results in many collisions or ability to find many pre-images and an attacker knows the procedure of your key, then it is possible that your data could have been tampered with, without use of your private key, by changing the data to a similarly hashed data but with different contents. Assuming that we are using a cryptographically sound encryption algorithm for our key, there is also a

possibility that someone might have taken our private key through malicious means and wrote an entirely new data and signature to the position *i*. When you come back to assess the data, you are accessing the new data with an entirely new signature, but since the data matches the signature from your key, you have no clue that your data has been affected.

Problem 5

Another approach would be hashing the concatenation of all records in the database. This hash is a small item that can be stored locally.

(a) What is the write/read/verify procedure for this system?

If you are writing to position *i*, then you will write the new data to the database and get returned a hash of the new hashing of the concatenation of the records which you store on your system. When you read from position *i*, you will pull the data from the database and verify it by the database giving you the current hash of the records and if that hash matches your previously stored hash, then the data will most likely not have been affected (unless that hashing is bad and people are able to find collisions in the encryption of the documents).

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

The hash returned will always be the same size no matter the scale of n because of the way that the hash works. The cost of both reading and writing in the database will go up with *n* because it will require the database to hash more and more data the more records that are written to it. While the hash length remains the same, the size of the database can grow and grow, which means that finding that hash will take longer for writing and reading.

Problem 6

Instead of using the concatenating all the records linear, they were organized into a Merkle tree.

(a) What is the write/read/verify procedure for this system?

If you are writing to position *i*, on the assumed right half of the Merkle tree, then the database will have to get the hash of left half of the Merkle tree, and the unaffected hashes in the right half of the Merkle tree. They would then have to recalculate the hashes from that new written position *i* up until the root is fixed to the new Merkle root and return that to the user. Assuming that you store just the Merkle tree root on your system, then all you have to do to read is ask for the data at position *i* as well as their current Merkle tree root. To verify you check your stored Merkle root with their current sent one and if they match then you can assume that your data has been unaffected (assuming once again that if the hashing isn't bad and doesn't have any easy collisions in the Merkle tree).

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

Because of the way that Merkle trees work, when writing and getting the Merkle tree, all you have to do is the height of the tree number of hashes. This is because the other hashes on one half will remain unaffected and anything that isn't directly associated with the changed hash does not need to be changed. This means that the cost of writing will be $\log_2(n)$ with n being the number of records on the database. For reading, all the database

has to do is return the Merkle tree root no matter the number of records. The cost will always be the same because the database will only be looking at the Merkle tree root.

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?

Assuming that all the nodes in network are honest (don't collaborate, aren't selfish, etc.), that the rate of blocks found is exactly 1 block every 10 minutes and that the hashing difficulty remains constant:

1440 minutes / day

144 blocks / day

21.6 blocks / day given α = 0.15

The mining pool would be expected to get around 21 to 22 blocks a 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.

$$\frac{t}{10m} * \alpha = blocks\ found\ in\ t\ minutes$$

Problem 8

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

$$\frac{L}{10m} * (1 - \alpha) * 22 = \text{orphaned blocks per day}$$

Where *L* is latency for 1 block sent by the mining pool, α is the hashing power of the mining pool.

Problem 9

How does this change if the mining pool is mining selfishly? (For this question, assume that the selfish mining pool learns of a block announced by the rest of the network as soon as it is announced, so will immediately announce any withheld blocks at that time. That is, you may still assume the simplified latency L model, but that the selfish mining pool has a spy in the other supernode with a low-latency direct connection to the mining pool.)

If the mining pool is mining selfishly then the orphaned blocks will be equal to the number of blocks that the selfish mining pool can get ahead of the supernode. This means that if the selfish mining pool in one day can mine ahead of the supernode a total of 4 times, then there will be a minimum of 4 orphaned blocks including the already possible orphan blocks based off of latency. However, the selfish mining pool would be extremely lucky to get many blocks ahead of the supernode but it is still possible.

Problem 10

In upcoming classes, we will have visits from a law professor (who also works for the State Department on international cyberlaw and promulgating the open Internet) and an FBI agent who works on criminals using bitcoin for ransom. Come up with at least one good question that you would like one of them to answer.

Law Professor Qs

What are your thoughts on the most recent initiatives to stop the open Internet?

How can you precisely determine if somebody was accessing illegal things if you don’t have photographic or physical evidence of it? For example, somebody could be hacked and the malicious attacker purchases an illegal good on that computer to be sent to that address, how in a court of law is someone found guilty, whether it’s the attacker or the victim?

How is it possible to enforce international cyberlaw? Do all countries commit to it?

FBI Agent Qs

What do you think about the viability of using Bitcoin for ransom?

Do you all try to track the ransomed Bitcoin if the ransom succeeds?

When did ransoming for Bitcoin become a noticeable occurrence?