Cody Robertson
(ctr3sf)
Problem set #2

Problem 1:

Satoshi's claim seems built on a few assumptions. The first is that the lack of response from the other peers in the Bitcoin network. Seeing that someone was reversing transactions, there could be an outright ban, or some other action, on any blocks that come from the attacker if they aren't more subtle than an outright "reversal" of funds. The claim also assumes that the currency would not be abandoned if enough people found it was being abused at such a level, leaving the attacker to be the king or queen of their respective metaphorical pile of disused digital coinage. This claim also assumes that the new coin generation in question is the altruistic mining to continue Bitcoin rather than greedy mining only one's own transactions to obtain a large sum of block discovery rewards.

On the more optimistic end, it assumes that the attacker would have a sizable enough lead in computing power to where the remaining power would be unable to recover in a reasonable amount of time. If the attacker were to pick an arbitrary position in the blockchain to try and create a fork to "steal back their payments", they must become the longest chain from that point. If the majority is slim enough, it may take the attack a long time to overtake the most current chain, making it unprofitable in a short term timescale as honest mining would be more immediately profitable with a majority of computing power.

Problem 2:

In English, this means that the probability of an attacker overtaking the current blockchain with a greedily mined fork is less than a tenth of a percent at a deficit of 340 blocks when the attacker controls a whopping 45 percent of the total Bitcoin mining power. The table as a whole gives a z value of a deficit of nodes and a q probability that an attacker finds their next block to give a final p probability of a greedy miner trying to overtake the current blockchain with their own mined fork.

Problem 3:

Calculated by a Python Implementation of the same algorithm implemented in the paper. See Appendix A

P < .05	
Q	**Z**
.10	3
.15	3
.20	5
.25	6
.30	10
.35	17
.40	36
.45	137

Cody Robertson
(ctr3sf)
Problem set #2

Problem 4:

a) Without information about the position in the bytes of data[i], you would only know that the information in that spot has been altered or unaltered, not if data is stored in different locations in the database than originally intended.
b) If the signature is created with an unexposed, computationally secure private key, you could check to if the original signature matches the computed signature of the data you just downloaded. The downloaded signature doesn't tell you if the data was altered in any way while in the cloud. Only successful match between these two signatures would tell you if the data was unaltered, as your private key could not be used by the cloud to resign an altered file.

Problem 5:

a) Procedure:
    a. Write – Concatenate all records, hash with a secure hash function, store the hash locally, and then upload all records.
    b. Read/Verify – Download all records. Compute the hash of the downloaded and concatenated records and compare that stored hash. If they match, then the data is what you uploaded. If they do not, then any number of records may have been tampered with.
b) The hash function will take longer and longer the larger n becomes, although this should not be a noticeable difference until extreme sizes of n. Because there is only one hash computed for a write and a read operation, they will scale by the same amount: n.

Problem 6:

a) Procedure:
    a. Write – Compute the hash of each record with a secure hash function, concatenate those hashes by pairs continuing until you are left with a Merkle root, store the Merkle root locally, and then upload all records.
    b. Read/Verify – Download all records. Begin to recompute the Merkel Root. If the two hashes match your data is whole and unaltered.
b) The hash function will take longer and longer the larger n becomes, although this should not be a noticeable difference until extreme sizes of n. Because you form a Merkle tree ever time for a write and a read operation, they will scale by the same amount: 2^{n+1}-1.

Problem 7:

a) 10 minutes per block is 6 blocks per hour. 6 blocks times 24 hours in a day is 144 blocks per day. If a mining pool has a 15 percent chance to mine a block, they can expect to mine 21.6 blocks on average per day, or just 21 blocks as one cannot mine a fraction of a block.

b) $E_{Blocks} = \alpha * \frac{t}{10}$ You would round down to the nearest whole number for a real world number of blocks in the timeframe of t.

Problem 8:

$$E_{Orphans} = \alpha * \frac{L}{600} * 144$$

Where 144 is the number of blocks per day and 600 is the amount of time in seconds it takes to mine a block.

Problem 9:

When mining greedily, the pool has one way to have their blocks orphaned: having the other supernode pulls ahead of the pool's supernode without extra blocks to reveal. Otherwise, the pool would publish their results for every node that other supernode finds to remain completive and have a lead or be equal. So the expected value of orphans for a greedy mining pool should be:

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

Problem 10b (The Easy Way Out):

- With the internet often being touted in media as a lawless "Wild West" stand in, is there really as little recourse as people assume there is for digital crime? Is it only when digital crime translates over to real world and established law that decisions can be made, like buying illegal drugs with Bitcoin?
- Does the global nature of the internet make it difficult to nail down cyber criminals or enact cyber law? Would legislation related to cyber law and regulations have to come from a single international point of authority or can each nation feasibly have their own set of rules that function well in the global landscape?
- Should cyber crime or cyber law just be thought of as ordinary law and/or crime? Is there a better term to use rather than cyber-____ that does not immediately bring 80s science fiction to mind?

Appendix A: Code for Question 3

```
#Cody Robertson
import math

def AttackerSuccessProbability(q,z):
    p = 1.0 - q
    lambda_value = z * (q/p)
    sum = 1.0
    k = 0
    while k <= z:
        poisson = math.exp(-lambda_value)
        i = 1
        while i <= k:
            poisson *= (lambda_value/i)
            i += 1
        sum -= poisson * (1 - math.pow(q/p, z-k))
        k += 1
    return sum

def FindZLessThanGivenPandQ(p_thres, q):
    z = 0
    p = 1
    print("P < " + str(p_thres))
    while p > p_thres:
        z += 1
        p = AttackerSuccessProbability(q,z)
        print("q=" + str(q) + " z=" + str(z) + "p=" + str(p))


def main():
    q=.05
    FindZLessThanGivenPandQ(q,.1)
    FindZLessThanGivenPandQ(q,.15)
    FindZLessThanGivenPandQ(q,.20)
    FindZLessThanGivenPandQ(q,.25)
    FindZLessThanGivenPandQ(q,.30)
    FindZLessThanGivenPandQ(q,.35)
    FindZLessThanGivenPandQ(q,.40)
    FindZLessThanGivenPandQ(q,.45)

if __name__ == '__main__':
    main()
```