Gardner Fiveash

Problem Set 2:

1. An attacker with more than 50% of the hashing power could either continue to mine new blocks and get the reward every 10 minutes, or use the power to divert the block chain and double spend any money he spent on the existing chain. However, this is public so people would lose trust in the currency and value would go down. So the assumptions are that people would be mad and notice if someone with more than 50% kept altering the block chain and stop receiving bitcoin for fear of accepting invalid transactions. People care about the stability of their currency, so they would pull their money out of bitcoin and the greedy attacker would lose money.

2. When an attacker is 340 blocks behind, there is less than a .1% chance the attacker with 45% of the hashing power will catch up

3. P < .05:

    a. q=0.10 z=3
    b. q=0.15 z=3
    c. q=0.20 z=5
    d. q=0.25 z=6
    e. q=0.30 z=10
    f. q=0.35 z=17
    g. q=0.40 z=36
    h. q=0.45 z=137

Code for #3.

```
#include <math.h>
#include <iostream>

using namespace std;

double AttackerSuccessProbability(double q, int z)
{
 double p = 1.0 - q;
 double lambda = z * (q / p);
 double sum = 1.0;
 int i, k;
 for (k = 0; k <= z; k++)
 {
 double poisson = exp(-lambda);
 for (i = 1; i <= k; i++)
 poisson *= lambda / i;
 sum -= poisson * (1 - pow(q / p, z - k));
 }
 cout << sum << endl;
 return sum;
}


int main() {
        AttackerSuccessProbability(.1, 3);
        AttackerSuccessProbability(.15, 3);
        AttackerSuccessProbability(.2, 5);
        AttackerSuccessProbability(.25, 6);
        AttackerSuccessProbability(.3, 10);
        AttackerSuccessProbability(.35, 17);
        AttackerSuccessProbability(.4, 36);
        AttackerSuccessProbability(.45, 137);
        return 0;
}
```

output:

```
0.0131722
0.0442278
0.0274155
0.0499426
0.0416605
0.0450402
0.0487612
0.0493437
```

4.
a. the problem is there's no indication of where the data your accessing is stored. All you know is its in the database as a whole.

b. Yes, beyond a reasonable doubt we are sure that this is the message. I'm assuming the process is concatenating your message with the private key and hashing the result to produce a signature. As long as the hash function we use is collision resistant, when we read back the signature it will be the same unless the data has been tampered with. However, I thought the whole point was that we don't have a local copy of the data itself so I don't see how we could compare signatures.

5.
Say you have a database with 10 messages. The read/write/verify procedure would be to hash the concatenation of all the messages, and then upload the messages into the database. You store the hash result locally. If you read the 5th message, you wouldn't know if it had changed or not. The only thing you can know is if the database has a whole has been altered, which you can find by reading all 10 messages and comparing the hash with the one you stored. So the cost scales by a

proportion of n since you have to read down every message to verify the whole database has been changed.

6.
For a merkle tree, the messages in your database are grouped in pairs, and each pair is hashed. The resulting hashes are then grouped in pairs and hashed until you get all the way to the top of the tree which is the root. You store the hash of the root locally. Say you want to verify the 2nd element. You would also need the first element, and the hashes of the other elements that get you to the merkle root. The resulting hash of the merkle root should be the same as the one you stored locally. The cost scales at logn, since you only need the shortest path of hashes to the root of the tree.

7.
a. 6*24 =144 blocks found in a day. .15(144) = 21.6 blocks.

b. $\alpha * (t/10)$

8.
I'm assuming that 'orphaned blocks for a mining pool' means the number of blocks the mining pool finds that gets orphaned.

Probability the pool finds a block * probality the rest of the network finds a block in latency period L * Probability the rest of the network adds a block onto the other chain and not yours

$144*\alpha \quad * \quad L*((1-\alpha) * 144) \quad * \quad ((1-\alpha) * 144)$

9. If the mining pool is selfish, then they don't publish when they find a block.

Now I'm assuming that I want to find how many blocks the selfish miners successfully orphan.

How many times you find two blocks before the rest of network finds one block (so you're always ahead).
$(144*\alpha)^2 - (1-\alpha) *144$

10 b.

Does the fact that the bitcoin currency was created by an unknown entity make you less likely to accept it as a mainstream currency? Similarly, what are the chances the US government ever prefers bitcoin to the dollar?