EE 374 Homework 2

Lucas Masterson August 10, 2025

1 Problem

Problem 1.a.

The random oracle function is an idealized abstraction of a hash function. That is, given an input pair (k_n, m), the hash function is expected to return a value that is uniformly random over the digest space T. It is also deterministic, meaning that same inputs result in same outputs.

Problem 1.b.

The network would be expected to take approximately 2^{80} attempts to mine this block.

Problem 1.c.

The collision resistance property states that for some key k, it is computationally infeasible to find two distinct messages m_0 and m_1 such that: $H(k_1, m_0) = H(k_1, m1)$. That is, different messages should not result in the same hash.

Problem 1.d.

The random oracle model, by definition, is collision resistant. The random oracle model is assumed to map values to a uniformly random digest space T deterministically, so the same inputs should result in the same outputs. If a hash function behaves like the random oracle model, then there would be a digest space of $\{0,1\}^{256}$, which gives a $1/2^{256}$ chance of finding a collision. This slim probability is negligible and computationally infeasible, thus giving that function the property of collision resistance.

2 Problem

The account model uses a ledger to assign a single entry per user, while the UTXO model uses a single entry per *coin*. Moreover, in the account model, each user is given a unique account which tracks balance and is modified by transactions. The UTXO model, however, allows users to own multiple coins and must refer to specific coins when spending. The account model is thus vulnerable to replay attacks, but the UTXO mitigates this by "consuming" coins, preventing them from being used in future transactions and splitting them into new coins in the process.

3 Problem

Problem 3.a.

We first begin with the two `OP_DUP` operations in the script, which duplicate the top item on the stack (7). The next `OP_ADD` command adds the two 7s together, yielding 14. This is then duplicated by `OP_DUP` and added (`OP_ADD`) with itself, yielding 28. The final `OP_ADD` adds 28 and 7, giving 35.

This script implements the function $f(x) = 5x$.

Problem 3.b.

The first command `OP_2DUP` duplicates the top two stack items, 3 and 11, respectively. This is followed by an `OP_MAX` which moves (`OP_TOALTSTACK`) the max of the two (11) to the alt stack. The next command (`OP_MIN`) takes the minimum (3) and subsequently pops the alt value from the top of the stack back into the main stack `OP_FROMALTSTACK`, giving 11 and 3.

This script implements $f(x,y) = [\max(x,y), \min(x,y)]$.

4 Problem

Source(s): [1]

```
version: 01000000
inputsCount: 01
input #1:
    txid: 8e70ee5afa829e776d006cdb340c79029e95d9bdd05975159019697172e944ea
vout: 00000000
scriptSigSize: 8a
scriptSig:
    OP_PUSHBYTES_71: 47
    data: 30440220694ff325724a4f4b0f3f0c36bf8e94cac58ad7c9b4
          d5bd8c7286c0da623f0b2c02206ae94680a8f31f30cd846da258e919
          c94afe2dd629b4f4ce11bbe8165ff99a5f01
    OP_PUSHBYTES_65: 41
    data: 04fc60372d27b067ca306ba812ced9c8cd69296b83a40b9b57c
          593258c1b9e0ee1c0c621ca558b878395f9645a4b67a96e51843e9c06
          0d43a3833fdd29a91f4f31
    sequence: ffffffff
outputsCount: 02
output #1:
    value: 00e1f50500000000
    scriptPubKeySize: 19
        scriptPubKey:
            OP_DUP: 76
            OP_HASH160: a9
            data: 14f369e8330a1e9a349721c3d790ae4d38b68e5252
            OP_EQUALVERIFY: 88
            OP_CHECKSIG: ac
    output #2:
        value: 00e9a43500000000
        scriptPubKeySize: 19
        scriptPubKey:
        OP_DUP: 76
        OP_HASH160: a9
        data: 14f10eb3bfce5ab24537e571ceb3862d2949f7c5e2
        OP_EQUALVERIFY: 88
        OP_CHECKSIG: ac
locktime: 00000000
```

5 Problem

Problem 5.a.

Given that *F2Pool* has 20% of the network's total hash rate, and that *F2Pool*'s mining successes are independent events, the time taken for *F2Pool* to mine their next block has an exponential distribution with rate $\lambda = 0.2 \cdot \frac{1}{600} = \frac{1}{3000}$.

The probability density function is $\frac{1}{3000} e^{-t/3000}$ and the mean is 3000 seconds.

Problem 5.b.

Since each mining event can be considered independent of each other (where each miner has an equal, random probability of finding a hash that scales with hash rate) and that *F2Pool* has 20% of the network's total hash rate, we can conclude that there is a 20% probability that th next block is mined by *F2Pool*.

Problem 5.c.

To successfully execute the attack, the attacker would have to mine 2 of the 3 blocks to gain the longest chain and thus overwrite the ledger. Supposing *F2Pool* has a hash rate that is 20% of the total network hash rate, the probability that the next block would be mined by *F2Pool* is $3 \cdot (0.2)^2 \cdot 0.8 = 0.096$.

The attackers would first have to add a transaction to B, then mine the subsequent 2 blocks to by making their private chain public. In this case, the attacker would have to mine all 3 blocks after the transaction enters the mempool. The probability that this attack will succeed is $3 \cdot (0.2)^2 \cdot 0.8 + (0.2)^3 = 0.104$.

References

[1] Robin Linus. Raw Bitcoin transactions hand-parsed for educational purposes — gist.github.com. https://gist.github.com/RobinLinus/7971daeb88ecd939825bc456721d9378. [Accessed 10-08-2025].