CS 251 Assignment 2

Luis A. Perez

Problem 1

Solution:

a. The block-chain will undergo a fork. The miners running implementation A will accept the transaction as valid, and since they own 80% of the mining power, they will simply continue etending this chain with the invalid transaction. The miners with implementation B however, will always see this longer chain as invalid, so they will simply continue working on the shorter (but valid) chain which does not contain the double spend.

b. If the situation were reversed, we would not have a fork. Since only 20% of the mining power would consider the chain with the double spend to be valid, we would expect that eventually the chain without the invalid transaction would win out, and all miner (even those what wanted to accepted the invalid transaction) would switch to the valid chain.

However, the 20% of miners would always have this double spend transaction which they are unable to put into a block (to have it be successfully permenantly accepted into the chain). You'd hope they would eventually notice this, and fix their software.

Problem 2

Solution:

a. Assuming 1 block every 10 minutes, we have 6 blocks/hour, which equates to roughly 75BTC/hour or US$450,000/hour (using 1BTC = US$6000). If we assume this entire reward is spent on electricity, at US$0.05/kWH, we'd be using 9,000,000 kWH in one hour.

b. With a difficulty of $D = 2^{75}$, we'd expect it to take 2^{75} hashes to solve a single block, which means in the span of an hour (to solve 6 blocks) we'd need approximately $6 * 2^{75}$ hashes. At $18 * 10^{12}$ hashes/sec, this would take 12,592,977,287.7 (12.5B) seconds or 3,498,049.24657 (3.5M) hours on a single Antminer S9 Hydro, consuming a total of 5,946,683.71917 (6M) kWH.

c. The primary reason is that miners to to be incentized to make a profit (so it is unreasonable to assume the entirety of the bitcoin reward is spent on electricity costs).

Problem 3

Solution: When an honest miner is aware of a fork in the chain, it should randomly select between the two chains which one to work on (uniformly so). This guarantees that honest miners won't be more likely to select S simply because of network connectivity, and would result in $\gamma \approx 0.5$. Furthermore, this change is backwards compatible since it does not affect the validity of either chain.

Problem 4

Solution:

a. We can simply take the expected fraction of winnings the pool achieves (α) and multiply it by the proportion of power the partipant provides to the pool $\frac{\beta}{\alpha}$. This will give us the expected fraction of overall mining rewards the miner will be.

$$\underbrace{\alpha}_{\text{Pool winnings}} \cdot \underbrace{\frac{\beta}{\alpha}}_{\text{Share of winnings}} = \underbrace{\beta}_{\text{Honest miner winnings}}$$

With the obvious restriction that $0 < \beta < \alpha < 1$.

b. We follow the same approach as above, however, using modified values. In this case, the expected fraction of overall mining rewards earned by the pool is $\frac{\alpha-\beta}{1-\beta}$, while the share of work done by the miner is still $\frac{\beta}{\alpha}$.

$$\underbrace{\frac{\alpha-\beta}{1-\beta}}_{\text{Pool winnings without miner helping}} \cdot \underbrace{\frac{\beta}{\alpha}}_{\text{Share of winnings}} = \underbrace{\frac{\beta(\alpha-\beta)}{\alpha(1-\beta)}}_{\text{Honest miner winnings}}$$

With similar restrictions where $0 < \beta < \alpha < 1$.

c. We follow a similar approach as to above. We first note that the total compute of the network dedicated to mining is $1 - \beta$ (since β proportion is being used by P_2 to sabotage P_1). There are two mechanism through which P_2 wins. The first is that P_1 solves the puzzle. This happens proportially to its mining power, which is given by $\frac{\alpha_2-\beta}{1-\beta}$. The other way to win is when P_1 wins, in which case only a share of the winnings is given to P_1. As such, we have that the fraction of the overall mining rewards is:

$$\underbrace{\frac{\alpha_2-\beta}{1-\beta}}_{P_2 \text{ winning share}} + \underbrace{\frac{\alpha_1}{1-\beta}}_{P_1 \text{ winning share}} \underbrace{\frac{\beta}{\alpha_1+\beta}}_{\text{Proportion from } P_1 \text{ given to } P_2}$$

With similar restrictions such that $0 < \beta < 1$ and $0 < \alpha_i < 1$ and $\beta < \alpha_2$.

d. If P_2 is honest, it will earn α_2 as rewards. As such, the attack from above is profitable

only when the following is satisfied:

$$\begin{aligned}
&\alpha_2 < \frac{\alpha_2 - \beta}{1-\beta} + \frac{\alpha_1\beta}{(1-\beta)(\alpha_1+\beta)} \\
\implies &(\alpha_2-\beta)(\alpha_1+\beta) + \alpha_1\beta > \alpha_2[\alpha_1 + (1-\alpha_1)\beta - \beta^2] \\
&\qquad\qquad \text{(Multiply through by } (1-\beta)(\alpha_1+\beta)) \\
\implies &\alpha_1\alpha_2 + \alpha_2\beta - \beta^2 > \alpha_1\alpha_2 + \alpha_2(1-\alpha_1)\beta - \alpha_2\beta^2 \quad \text{(Expanding terms)} \\
\implies &\beta(\alpha_2-\beta) > \beta[\alpha_2(1-\alpha_1) - \alpha_2\beta] \quad \text{(Simplify)} \\
\implies &\alpha_2 - \beta > \alpha_2(1-\alpha_1) - \alpha_2\beta \quad (\beta \neq 0) \\
\implies &(1-\alpha_2)\beta < \alpha_1\alpha_2 \\
\implies &\beta < \frac{\alpha_1\alpha_2}{1-\alpha_2}
\end{aligned}$$

As such, given two pools P_1 and P_2 each with mining power α_1 and α_2 respectively, sabotaging P_1 is profitable for P_2 only if $\beta < \frac{\alpha_1\alpha_2}{1-\alpha_2}$ when compared to honest mining. We can provide concrete values. For example, we could have:

$$\begin{aligned}
\alpha_1 &= 0.4 \\
\alpha_2 &= 0.2 \\
\beta = 0.099 < \frac{0.4 * 0.2}{1-0.2} &= 0.1
\end{aligned}$$

We can verify by plugging these values into the equation from (c), to see that the expected reward under this strategy for P_2 is given by:

$$\frac{0.2-0.05}{1-0.05} + \frac{0.05}{0.4+0.05}\frac{0.4}{1-0.05} \approx 0.204678 > 0.2$$

And as such, using this strategy, P_2 will expect to take ever so slightly more winning that if he plays honestly.

Problem 5

Solution:

a. We argue that if all players are honest (following the protocol as described) and that $\delta > \Delta$, the at the beginning of any timestep t all players have perfectly consisten view of the longest chain, except with a very small probability.

To argue this, let us first establish that at the beginning of timestep t_0, all players have a consist view of the longest chain (namely, that the chain is empty).

For an inductive argument, let us now assume that at timestep t_i, all players have a consistent view of the longest chain (eg, they all agree that the longest chain is the same). Then at time t_i, all players j such that $\mathcal{H}(PK_j, t_i) < \frac{2^\lambda}{N}$ will be *eligible* players. Since all players are honest, only these players will broadcast their respective blocks:

$$B_j = (h_{-1}, \texttt{txs}_j, t_i, PK_j)$$

and **no other** blocks will be broadcast. Note that, by assumption, all players agree on h_{-1} and on t_i. They may, however, have different $\texttt{txs}_j$, but since all players are honest, all transactions will be valid. As such, all players receiving blocks B_j will accept these blocks as valid extensions of their view of the block-chain.

By the properties of network delay, we can guarantee that by $t = t_i + \Delta$, all players in the network will have received all eligible blocks B_j. As such, each player will have a locally forked chain (forks at the last block) if there were multiple eligible players. All honest players will chose to keep the fork with the most recent timestamp (which fails in this case), and failing that, will keep the block with the lower eligibility hash $\mathcal{H}(PK_j, t_i)$. By the properties of hash functions, there is negligible probability for a collision between the eligibility hashes, and as such, with extremely high probability, all honest players will deterministically agree on the block j with the lowest eligiblity hash.

As such, we can argue that by time $t = t_i + \Delta$, all honest players have a consist view of the longest chain. Since $\Delta < \delta$, then we know that all honest players have a consisten view of the longest chain by time $t = t_i + \Delta < t_i + \delta = t_{i+1}$, concluding our inductive argument.

b. If we modify the protocol so that the entire block tuple is used to determine eligibility, we run into the problem that $\mathcal{H}(h_{-1}, txs, t, PK)$ is no longer unique at time t for player with key PK. Since this is not unique, a dishonest player *can make himself always eligible with high probability* and *can force other nodes to accept his block.*

To do this, the dishonest player simply spends his time trying out slightly tweaked txs (eg, modifying the nonce within a transaction, for example) computing m values

of $\mathcal{H}(h_{-1}, txs_i, t, PK_{\text{dishonest}})$ for $i = 1, 2 \cdots, m$. Then at time t, the dishonest player simply broadcassts the block:

$$B_{\text{dishonest}} = \min_i \{\mathcal{H}(h_{-1}, txs_i, t, PK_{\text{dishonest}})\}$$

With this strategy, the probability that the dishonest player has the lowest eligibility hash is simply given by, where i are honest:

$$P[\forall i, \mathcal{H}(B_{\text{dishonest}} < \mathcal{H}(B_i))] = \frac{m}{N+m}$$

As such, we see that for sufficiently large m, the dishonest player's block will be the one used to extend the chain (since it will have the lowest eligibility hash), except with very small probability. In this way, the dishonest player can force all other honest players to accept his block, with high probability, and thereby destroy the liveness property of the protocol (the only player who can add transactions is the dishonest player).

c. The easiest way for the longest valid chain at time t' to omit h is if the adversary managed to extend the longest fork in C_t omitting h, let us call this chain F_t, to be `longer` than the chain owned by the honest players. As such, this can happen if and only if at some time t', $|C_{t'}| \geq |F_{t'}|$ (this probability of this event therefore provides an upper bound on the probability that the honest player's view omits h). To compute this probability, let us consider $t' = t + m$ (m timestamps have passed without the adversary overtaking the main chain). At thsi point, let us compute the probability that the adversary overtakes the main chain. First, note that we have the following:

$$\begin{aligned}
|F_{t'}| &= |F_t| + X_m && (X_m \text{ as given in the problem}) \\
|C_{t'}| &= |C_t| + Y_m && (Y_m \text{ as given in the problem}) \\
|C_t| &= |F_t| + k && (\text{As given in the problem})
\end{aligned}$$

We want to compute $\Pr[|F_{t'}| \geq |C_{t'}|]$, which is given by:

$$\begin{aligned}
\Pr[|F_{t'}| \geq |C_{t'}|] &= \Pr[|F_t| + X_m \geq |C_t| + Y_m] \\
&= \Pr[|C_t| - k + X_m \geq |C_t| + Y_m] && (\text{Simplifying}) \\
&= \Pr[X_m - Y_m \geq k] && (\text{Re-arranging terms}) \\
&< e^{-m} \frac{1-p}{1-2p} \left(\frac{p}{1-p}\right)^k \\
&< \frac{1-p}{1-2p} \left(\frac{p}{1-p}\right)^k && (\text{Maximized for } m = 0 \implies e^{-m} = 1)
\end{aligned}$$

Note that for $p < \frac{1}{2}$, we have that:

$$\frac{p}{1-p} < 1$$

As such, we have have that the probability decays exponentially in k, which means it is vanishingly small.

d. If honest miners started accepting future timestamps, this could easily be abused by an adversary to guarantee that only blocks from the adversary are included in the chain (thereby preventing liveness). Suppose $t_{e_1}, t_{e_2}, \cdots, t_{e_k}$ are the timesteps during which the adversary is an eligible players. Then at $t = t_{e_i}$, the adversary publishes a block whose timestamp is $t = t_{e_{i+1}}$. Note that this block will be the accepted block, since it is the one with the most recent timestemp (all other blocks will have a timestamp of t_{e_1}).

However, we now consider what happens when an honest miner attempts to extend this chain. The protocol indicates that h_{-1} is the hash of a previous block with an "earlier" timestamp. As such, h_{-1} cannot point to $t_{e_{i+1}}$ (for an honest miner), and as such, the honest miner is forced to fork the chain from an earlier point. However, this chain is now either shorter (in which case it is ignored) or the same length as the adversary's chain but has a less recent timestamp (so also ignored). As such, this has the effect that honest miner's cannot publish block until time $t_{e_{i+1}}$. However, at this time, the adversary is once again eligible, and he can simply publish a block with a timestamp equal to his next eligibility time.

In this way, the chain will only ever contain the adversary's blocks, thereby preventing liveness.

e. The adversary can take advantage of the current protocol to prevent liveness by simply shutting down the players which are eligible at timestep t. We note that the eligibility of the current protocol is satisfied if for player i the inequality:

$$\mathcal{H}(PK_i, t) < \frac{2^\lambda}{N}$$

holds. This inequality is fully deterministic and can be computed by the adversary ahead of time. As such, the adversary knows which players will be eligible at time t. If he can shut down those players for at least δ time, he will prevent them from broadcasting any blocks, and thereby prevent liveness of the protocol (since transactions to honest players will never post).

Let p be the fraction of players that the adversary can shutdown or control. The adversary will fail if and only if the number of eligible players, N_e, is greater than pN (the number of players he can shutdown). However, N_e is a binomial random variable with $n = N$ and $p = \frac{1}{N}$. As such, it's expected value is 1, and the probability that it's larger than pN decreases is extremely small for reasonable values of p (eg, $p > \frac{1}{N}$)).

f. If we use a VRF instead of the given hash function, we can solve the problem in (e). We redefine the eligibility protocol such that players are eligible at time t when the following condition holds:

$$\mathcal{F}(SK_i, t) < \frac{2^\lambda}{N}$$

where $\mathcal{F}$ is a verifiable random function and SK_i is a secret held by player i (and which only player i knows). Honest players will then broadcast the output of this function along with the block they are proposing (if they are eligible to propose). Other honest players can then verify a block comes from an eligible players by simply verifying that:

$$\text{Verify}(PK_i, t, F(SK_i, t)) == 1$$

In this way, an adversary has no way of knowing which players are eligible ahead of time (he will only know once they've broadcast their block), and as such the adversary can at best shut down a fraction p less that $\frac{1}{2}$ players at random. However, this will succeed with only probability p.

g. We would adapt the protocal to the weighed PKI model as follows (we assume the question wants to adapt the original protocol, though note it is trivial to also implement the fix proposed in (f)). Let each PK_i have weight w_i associated with it, and let $W = \sum_i w_i$. Then they are eligible to propose a block if and only if:

$$\mathcal{H}(PK_i, t) \leq \frac{2^\lambda w_i}{WN}$$

Note that the probability that PK_i is eligible is now given simply by $\frac{w_i}{N}$, rather than just $\frac{1}{N}$.

Problem 6

Solution: The issue with Bob's contract is that it makes use of `tx.origin`. Suppose Mallory can trick Bob into calling a method on a contract she controls. When Bob calls this method, `tx.origin` will be Bob's address. Since Mallory controls this contract, she can have her contract call `BobWallet.pay` using Mallory's address as the destination, with whatever amount she wants as the amount.

When Bob's contract is called by Mallory's, `tx.origin` will be Bob's address, since this is the user address from which the transaction chain originally started. As such, Bob's contract will validate appropriately, and the money will be sent to Mallory's account.