Introduction to Cryptocurrencies (67513)
Exercise 5

Alon Netser 31160253-6
Daniel Afrimi 20386583-7

June 6, 2020

1 Selfish Mining

In class we saw the description of a "Selfish Mining" attack for a miner with an α-fraction of the hash rate and a γ-fraction of the "distribution power" in the network.

Question 1. Show that the fraction of rewards an attacker of power α can obtain is upper bounded by $\frac{\alpha}{1-\alpha}$.

Solution. Note that an attacker can not do better than using each one of his blocks to override one block of the honest nodes. This means that each one of the blocks mined by the attacker is used to eliminate a block of the honest nodes.

Formally, we'll assume the process is determined in discrete time-steps $1, 2, \ldots, T$, where each time-step is the creation of one block in the overall network (both by the attacker and by the honest nodes). Denote by $\ell_a^{(T)}$ the amount of blocks the attacker mined up to time T, and $\ell_h^{(T)}$ the amount of blocks the honest nodes mined up to time T. Note that $\ell_a^{(T)} + \ell_h^{(T)} = T$ (since the total amount of blocks are T). Denote by $r_a^{(t)}$ and $r_h^{(t)}$ the reward that the attacker and the honest nodes gained at time $1 \leq t \leq T$, respectively.

Note that the honest nodes "loses" reward every time the attackers eliminated one of their blocks, and we know that the number of honest-nodes blocks that were eliminated is bounded by the number of blocks that attacker mined (because the attacker needs to publish a longer chain than the honest nodes' chain in order to eliminate their blocks). Therefore we can conclude that the amount of "loosed" blocks of the honest nodes at time T which is $\ell_h^{(T)} - \sum_{t=1}^{T} r_h^{(t)}$ (the reward they were "supposed" to get is $\ell_h^{(T)}$ which is 1 for each block they mined, but they got $\sum_{t=1}^{T} r_h^{(t)}$ instead) is upper bounded by $\ell_a^{(T)}$, so $\ell_h^{(T)} - \sum_{t=1}^{T} r_h^{(t)} \leq \ell_a^{(T)}$.

Now we calculate the expected fractional reward (fraction of the rewards the attacker get, among the reward all of the network get) for the attacker following any policy π he please, when the time T approaches infinity. We get that

$$\rho = \lim_{T \to \infty} \frac{\sum_{t=1}^{T} r_a^{(t)}}{\sum_{t=1}^{T} r_a^{(t)} + \sum_{t=1}^{T} r_h^{(t)}}$$

Using the previously claimed inequality $\ell_h^{(T)} - \sum_{t=1}^{T} r_h^{(t)} \leq \ell_a^{(T)}$ we get that

$$\rho \leq \lim_{T\to\infty} \frac{\sum_{t=1}^{T} r_a^{(t)}}{\sum_{t=1}^{T} r_a^{(t)} + \ell_h^{(T)} - \ell_a^{(T)}} = \lim_{T\to\infty} \frac{1}{1 + \frac{\ell_h^{(T)} - \ell_a^{(T)}}{\sum_{t=1}^{T} r_a^{(t)}}}$$

(In the inequality we decreased the denominator and therefore increased the overall term, and in the equality we divided both numerator and denominator by $\sum_{t=1}^{T} r_a^{(t)}$). Now we use the fact that the sum of the rewards the attacker got (i.e. $\sum_{t=1}^{T} r_a^{(t)}$) is upper bounded by the amount of blocks he mined (i.e. $\ell_a^{(T)}$). Therefore we get that

$$\rho \leq \lim_{T\to\infty} \frac{1}{1 + \frac{\ell_h^{(T)} - \ell_a^{(T)}}{\ell_a^{(T)}}} = \lim_{T\to\infty} \frac{\ell_a^{(T)}}{\ell_h^{(T)}}$$

Now we finished our argument, since when $T \to \infty$ then $\ell_a^{(T)} \to \alpha \cdot T$ and $\ell_h^{(T)} \to (1-\alpha) \cdot T$ so the above limit equals $\frac{\alpha}{1-\alpha}$ as required. □

Question 2. Show that when $\gamma = 1$, this bound can be achieved, and describe a strategy that achieves the bound.

Solution. We note that in the previous question calculation we had mostly equalities, and only two inequalities. The first inequality follows from the fact that the amount of blocks "wasted" by the honest-nodes is at most $\ell_a^{(T)}$ (i.e. the number of blocks the attacker mined). When $\gamma = 1$ the attacker can use any one of his blocks to override a honest-nodes' block. He can do so by saving his block private, and whenever the honest nodes publishes a block, he'll publish his first block in his private chain quick enough so it'll reach all nodes in the network, causing the honest-nodes' block to be discarded. Therefore by using this strategy (which is done in the original selfish-mining paper) the first inequality is actually equality.

Regarding the second inequality, it is due to the fact that some of the attacker's block might be discarded (and then $\sum_{t=1}^{T} r_a^{(t)} < \ell_a^{(T)}$). However, when $\gamma = 1$ the attacker can not lose blocks at all, because every block he mines is published to all of the network (as soon as the honest-nodes publish a single block) fast enough so it will always replace the honest-nodes' block, so none of the attacker's blocks are discarded, and the second inequality is also an equality.

Overall, we got that when $\gamma = 1$ the bound $\frac{\alpha}{1-\alpha}$ can be achieved using the strategy from the original selfish-mining paper. □

2 Block-Withholding attacks

Assume that there is a large mining pool P that is "open", meaning that anyone can join it. Consider the following attack - an attacking miner joins the pool. Whenever the attacker finds a share, she sends it to the pool, but when she finds a block, she does not share it. The pool thus shares its revenue with the attacker when others find blocks, but does not gain from the attacker's work. We call this a "Block Withholding" attack. Notice that the attacker does not gain here in relation to mining in the pool honestly, but does hurt the pool.

We turn to a special case of the block-withholding attack - block withholding between pools. Assume there are two pools. P_1 is a malicious pool, and P_2 is an honest pool (the victim). Assume that P_1 decided to use the following attack: Some of the miners in P_1 join P_2. They submit shares, but they never publish a block if they find one. Let H_1, H_2 be the mining power of P_1, P_2 respectively. Let α be the fraction of miners (with hash rate $\alpha \cdot H_1$) that are infiltrating P_2 in order to attack it. Let H be the total mining power that exists outside of the two pools (all other miners).

Question 3. We define the utility U of a pool to be the expected reward it collects per second from mining (we ignore electricity costs, and other costs). Recall that the rewards given out to each pool are proportional to the number of blocks it places on the chain, and that pools divide rewards between miners proportionally to their mining power. Write an expression for the revenue of each of the pools if a block withholding attack is taking place.

Solution. Note that the amount of hash-rate used by the whole network to mine blocks is $(1-\alpha)\cdot H_1 + H_2 + H$. This is because a fraction α of the miners in the pool P_1 do not publish their blocks so they are "wasting" their computing power (with respect to block rewards, but they still gain from shares in P_1).

Now, the hash-rate of the pool P_1 is $(1-\alpha)\cdot H_1$, so the expected reward for P_1 for blocks mined by its own pool is

$$\frac{(1-\alpha)\cdot H_1}{(1-\alpha)\cdot H_1 + H_2 + H}$$

However, note that the pool P_1 profits also from blocks mined by the pool P_2, since they publish their shares there. The mining power of the pool P_2 used to mine blocks (which excludes the miner of P_1 that moved to P_2 because they do not publish their blocks) is H_2, and the pool P_1 will get a revenue of $\frac{\alpha\cdot H_1}{\alpha\cdot H_1 + H_2}$ fraction of the rewards for blocks mined by P_2, so overall the expected reward for P_1 from blocks mined by P_2 is

$$\frac{\alpha\cdot H_1}{\alpha\cdot H_1 + H_2}\cdot\frac{H_2}{(1-\alpha)\cdot H_1 + H_2 + H}$$

so overall the expected reward for the pool P_1 is

$$U_1 = \frac{(1-\alpha)\cdot H_1}{(1-\alpha)\cdot H_1 + H_2 + H} + \frac{\alpha\cdot H_1}{\alpha\cdot H_1 + H_2}\cdot\frac{H_2}{(1-\alpha)\cdot H_1 + H_2 + H}$$

As for the pool P_2, we mentioned previously that their expected number of blocks per second is $\frac{H_2}{\alpha\cdot H_1 + H_2 + H}$, and the (honest) miners in P_2 are a fraction of $\frac{H_2}{\alpha\cdot H_1 + H_2}$ of the total

amount of miners in that pool. So overall the expected reward of the pool P_2 is

$$U_2 = \frac{H_2}{\alpha \cdot H_1 + H_2} \cdot \frac{H_2}{(1-\alpha) \cdot H_1 + H_2 + H}$$

□

Question 4. Assume that $H_1 = \frac{1}{4}, H_2 = \frac{1}{3}$. What is the optimal attack strategy for P_1? Plot the attacker's utility for varying values of α.

Solution. In the previous question we calculated the expected reward for the pool P_1 given α as an argument

$$U_1(\alpha) = \frac{(1-\alpha)H_1}{(1-\alpha)H_1 + H_2 + H} + \frac{\alpha H_1}{\alpha H_1 + H_2} \cdot \frac{H_2}{(1-\alpha) \cdot H_1 + H_2 + H}$$

We'll plug-in the given values $H_1 = \frac{1}{4}, H_2 = \frac{1}{3}, H = 1 - \frac{1}{3} - \frac{1}{4} = \frac{5}{12}$ and get

$$\begin{aligned}
U_1(\alpha) &= \frac{(1-\alpha)\frac{1}{4}}{(1-\alpha)\frac{1}{4} + \frac{1}{3} + \frac{5}{12}} + \frac{\alpha\frac{1}{4}}{\alpha\frac{1}{4} + \frac{1}{3}} \cdot \frac{\frac{1}{3}}{(1-\alpha)\frac{1}{4} + \frac{1}{3} + \frac{5}{12}} \\
&= \frac{\frac{1}{4} - \frac{1}{4}\alpha}{1 - \frac{1}{4}\alpha} + \frac{\alpha\frac{1}{4}}{\alpha\frac{1}{4} + \frac{1}{3}} \cdot \frac{\frac{1}{3}}{1 - \frac{1}{4}\alpha} \\
&= \frac{\frac{1}{4} - \frac{1}{4}\alpha}{1 - \frac{1}{4}\alpha} + \frac{\frac{1}{12}\alpha}{-\frac{1}{16}\alpha^2 + \frac{1}{6}\alpha + \frac{1}{3}} \\
&= \frac{1-\alpha}{4-\alpha} + \frac{4\alpha}{-3\alpha^2 + 8\alpha + 16}
\end{aligned}$$

We'll now derive this function and compare to 0, in order to find its maximum (and we'll prove that it's indeed a maximum and not a minimum or a saddle point). The derivative is due to WolframAlpha :)

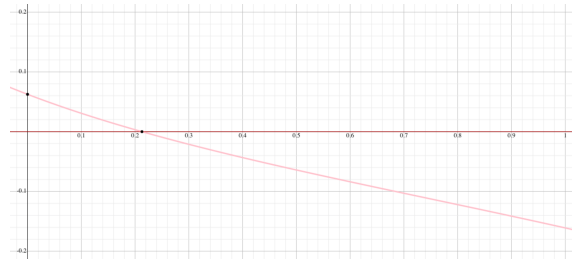
$$U_1'(\alpha) = \frac{-15\alpha^2 - 72\alpha + 16}{(3\alpha+4)^2 \cdot (4-\alpha)^2}$$

Using WolframAlpha to calculate the roots, and Symbolab to plot the graph we get that the two roots are $-\frac{12}{5} - \frac{16\sqrt{\frac{2}{3}}}{5} \approx -5.01$ and $\frac{16\sqrt{\frac{2}{3}}}{5} - \frac{12}{5} \approx 0.21$. Figure 1 shows the graph of the derivative. Clearly, the derivative is positive when $0 < \alpha < \frac{16\sqrt{\frac{2}{3}}}{5} - \frac{12}{5}$ and negative when $\frac{16\sqrt{\frac{2}{3}}}{5} - \frac{12}{5} < \alpha < 1$ so overall we get that this is a maximum of the function U_1 in the range $[0, 1]$.

Overall, we got that the attacker optimal strategy is to move $\alpha = \frac{16\sqrt{\frac{2}{3}}}{5} - \frac{12}{5} \approx 0.21$ of the miners in P_1 to P_2, and this will maximize its expected reward. Figure 2 shows the utility U_1 as a function of $\alpha \in [0, 1]$. Note that the utility in $\frac{16\sqrt{\frac{2}{3}}}{5} - \frac{12}{5} \approx 0.21$ is indeed the maximum, and in particular it's larger than when $\alpha = 0$, so the pool P_1 actually benefits from this block-withholding attack.

□

(a) The graph of U_1'.



(b) U_1' zoomed-in to the range $[0, 1]$.

Figure 1: The graph of the derivative of the U_1 as a function of α. Note that the two roots are $-\frac{12}{5} - \frac{16\sqrt{\frac{2}{3}}}{5} \approx -5.01$ and $\frac{16\sqrt{\frac{2}{3}}}{5} - \frac{12}{5} \approx 0.21$.

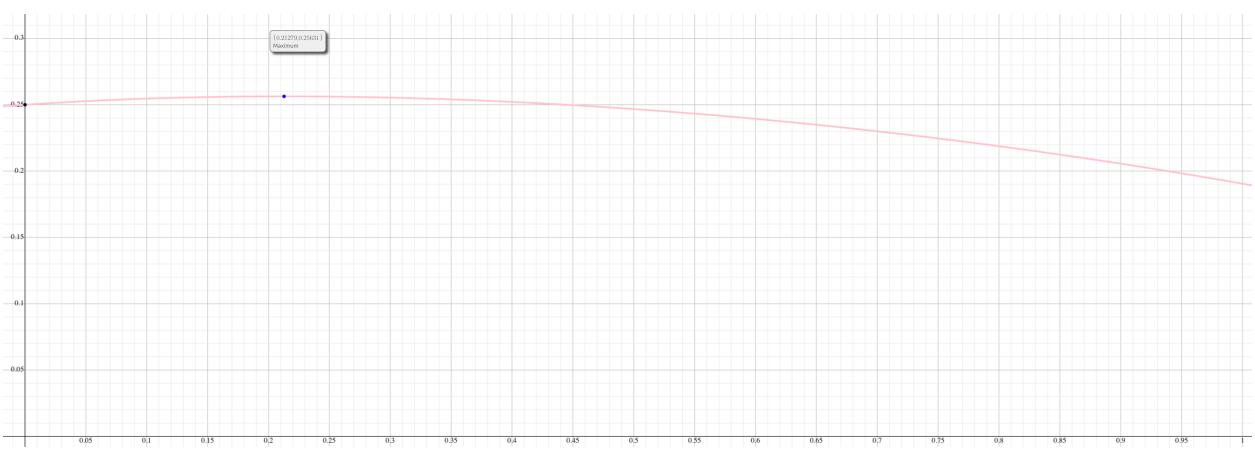


Figure 2: The utility U_1 as a function of α, for $\alpha \in [0, 1]$.

3 Pool Hopping

Assume some pool is using the following reward scheme to pay its workers: While a block is valid if $hash(header) < Target$, a share is valid if $hash(header) < k \cdot Target$.

Given that miners submitted n shares between blocks found by the pool, the rewards are distributed evenly between the shares. i.e., each share receives R/n of the reward (the block itself is also considered a share for this purpose). We saw that this payment scheme is susceptible to pool hopping, i.e., miners will leave the pools after several shares have been submitted.

Question 5. Write an expression for the expected reward of the i'th share submitted to the pool.

Solution. Let's denote the probability of generating a hash which is smaller than the target by p, so $p = \Pr[hash(header) < Target]$. Let's look at the probability that a single share is below the actual target, i.e. $\Pr[hash(header) < Target \mid hash(header) < k \cdot Target]$. Assuming the hash-function distributes uniformly over its range, it follows that this probability is exactly $\frac{1}{k}$ (which is the fraction of the range $[0, k \cdot Target)$ that is smaller than $Target$). We'll denote $q = \frac{1}{k}$.

Let's demote by X the random-variable which is the number of shares in the round. Note that the probability that there will be N shares in a round (i.e. the first $N-1$ hashes are below $k \cdot Target$, and the N'th share is below $Target$ is distributed geometrically with parameter q, so

$$\Pr[X = N] = q \cdot (1-q)^{N-1}$$

(the previous $N-1$ shares were not below $Target$, and this happens with probability $1-q$, and the N'th share was indeed below $Target$).

Now, let's focus on writing an expression for the expected reward of the i'th share submitted to the pool. When the i'th share is submitted to the pool, there are already $i-1$ shares in it. This means that we know that there number of shares in the round is greater then $i-1$. Formally, this means that we are conditioning of the event that $X > i-1$. Now, there are many (actually, infinite) possibilities for X, it could be i (meaning that the i'th share will form a block), it could be $i+1, i+2$, etc. Overall, when $X = N$ the reward will be R/N. Overall we can write that

$$\begin{aligned}
\mu &= \sum_{N=0}^{\infty} \frac{R}{N} \cdot \Pr[X = N \mid X > i-1] \\
&= \sum_{N=0}^{i-1} \frac{R}{N} \cdot \overbrace{\Pr[X = N \mid X > i-1]}^{0} + \sum_{N=i}^{\infty} \frac{R}{N} \cdot \Pr[X = N \mid X > i-1] \\
&= \sum_{N=i}^{\infty} \frac{R}{N} \cdot q \cdot (1-q)^{N-1-(i-1)}
\end{aligned}$$

Remembering that $q = \frac{1}{k}$ we get an expression for the expected reward for the i'th share:

$$\mu = \sum_{N=i}^{\infty} \frac{R}{N} \cdot \frac{1}{k} \cdot (1 - \frac{1}{k})^{N-i}$$

$\square$

Question 6. Assume that $k = 10^4$. After how many shares is it more profitable to mine solo than to participate in the pool? Show how you determined the number of shares.

Solution. Plugging-in $k = 10^4$ in the result form the previous question, we get that the expected reward of the i'th share is

$$\mu = \sum_{N=i}^{\infty} \frac{R}{N} \cdot \frac{1}{10^4} \cdot (1 - \frac{1}{10^4})^{N-i}$$

Instead of participating in the pool, the miners can mine solo and try to create blocks by themselves. In this case, when a miner create a share, i.e. he found a hash that is smaller than $k \cdot Target$, there is a probability of $\frac{1}{k}$ that this hash is in fact below $Target$, and then his reward will be R (he will keep it all for himself). Therefore, the expected reward for the share when mining solo is $\frac{1}{k} \cdot R = \frac{R}{10^4}$, v.s. the expected reward of the share when participating in the pool (and there are already $i-1$ shares in it) which is μ. Thus, if $\mu < \frac{R}{10^4}$ then it is more profitable to mine solo. Note that

$$\begin{aligned} \mu < \frac{R}{10^4} &\iff \\ \sum_{N=i}^{\infty} \frac{R}{N} \cdot \frac{1}{10^4} \cdot (1 - \frac{1}{10^4})^{N-i} < \frac{R}{10^4} &\iff \\ \sum_{N=0}^{\infty} \frac{1}{N+i} \cdot (1 - \frac{1}{10^4})^{N} = \sum_{N=i}^{\infty} \frac{1}{N} \cdot (1 - \frac{1}{10^4})^{N-i} < 1 \end{aligned}$$

We write a simple python script to calculate these infinite sums, for many values of $i \in \{1, 2, \dots\}$ to find out what is the smallest i for which the infinite sum

$$\sum_{N=0}^{\infty} \frac{1}{N+i} \cdot (1 - \frac{1}{10^4})^{N}$$

is smaller than 1. In order to calculate an infinite-sum, we require a desired approximation of $\epsilon = 10^{-49}$ which means that for each infinite-sum, we are actually calculating a finite sum where each of the values in the infinite tail is less than ϵ. Note that we chose ϵ by increasing the amount of variables we sum, from $10^3, 10^4, \dots$ until we reached 10^6 which gave us $\epsilon = 10^{-49}$ which seems reasonable enough.

Overall we found that for $i = 4348$ the infinite-sum is 0.99998, whereas for $i = 4347$ the result is 1.00011, so we conclude that **after 4348 shares it is more profitable to mine solo that to participate in the pool.** The code and the plot of the infinite-sums per $i \in \{1, \dots, 10000\}$ are in the next page.

```python
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm

# Approximation guarantee (in order to "sum an infinite sum",
# we require the last element to be less than EPSILON.
EPSILON = 1e-49

# The infinite sum will be summed up to MAX_N,
# guaranteed that its last elements will be less than EPSILON.
MAX_N = 10 ** 6

# The maximal i to check its result.
MAX_I = 10000

k = 10 ** 4  # A share will be valid if it's at most k multiplied by the target.
r = np.arange(MAX_N)  # Will be used for the summation.
mutual_arr = ((1 - (1 / k)) ** r)  # Will be used for the summation.

results = np.empty(MAX_I)

for j in tqdm(range(1, MAX_I+1)):
    arr = (1 / (r + j)) * mutual_arr
    assert arr[-1] < EPSILON, "The approximation is not good enough"
    results[j-1] = np.sum(arr)

i = np.where(results < 1)[0][0]
plt.scatter(i, results[i])
plt.axhline(y=1, color='gray', linestyle='--')
plt.axvline(x=i, color='gray', linestyle='--')
plt.ylim(0, 3)
plt.plot(results)
plt.savefig('q3.svg')
plt.show()

print(f"For i={i} the result is {results[i]:.5f}, "
      f"whereas for i={i-1} the result is {results[i-1]:.5f}")
```

□

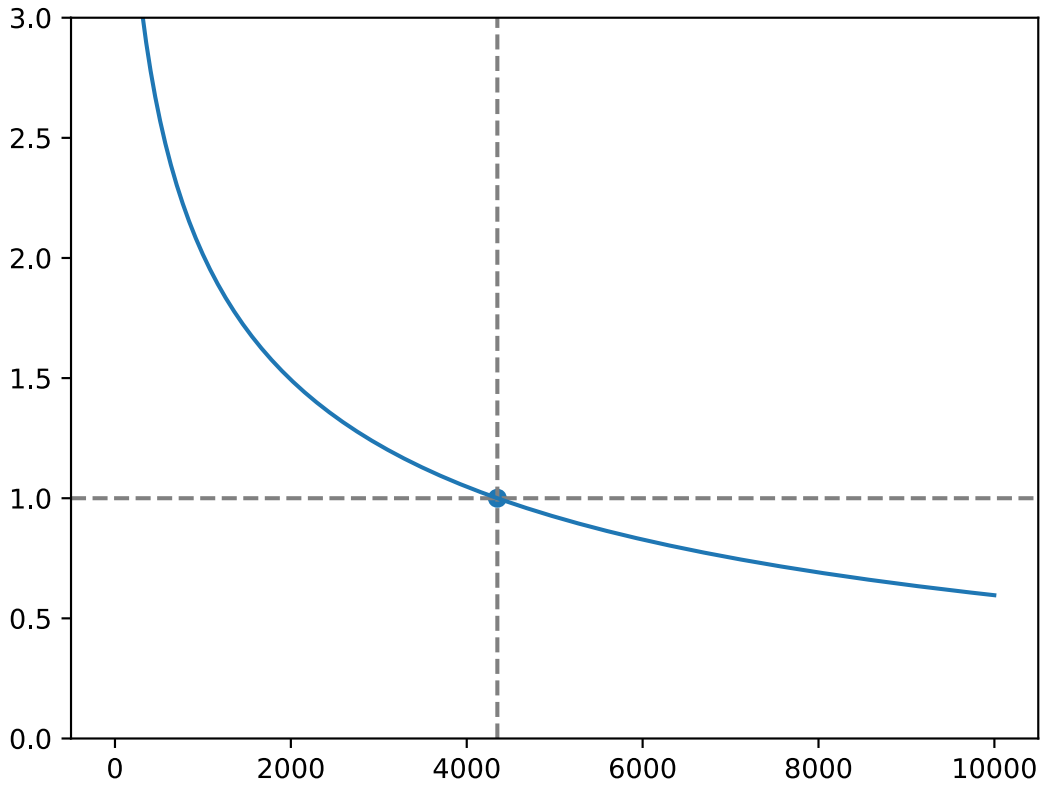


Figure 3: The infinite-sum per i.