**EE374 Internet-scale Consensus** **Stanford, Spring 2025**

# Homework #3 Solution

Due: Friday, April-25-2025, 11:59pm – Gradescope entry code: `R57ZN7`

Please upload your answers timely to Gradescope. Start a new page for every problem. We strongly suggest LaTeX to type your answers. For the programming/simulation questions you can use any reasonable programming language (please no assembly, brainfuck, etc. ☺). Comment your source code and include the code and a brief overall explanation with your answers. A tentative point distribution (in % of the total) is provided in brackets. For most problems there is more than one valid way of solving them!

1. (40%) We did a partial analysis of Nakamoto's private attack on the $k$-deep confirmation rule in lecture 3, and we will complete it in this problem. As in the lecture, the attack begins at time 0, and it is on the first honest block $b$, at level 1. The adversary has a fraction $\beta < 1/2$ of the mining power, the honest and adversary miners mine at rate $\lambda_h$ and $\lambda_a$ blocks per second respectively, and the network delay is assumed to be 0.

   a) What is the relation between $\beta, \lambda_h$ and $\lambda_a$?

   ---

   **Answer:**

   The adversary's mining rate is $\lambda_a$, and the honest mining rate is $\lambda_h$. $\beta$ is the fraction of the total mining rate that the adversary has. Hence;

   $$\beta = \frac{\lambda_a}{\lambda_a + \lambda_h}.$$

   ---

   b) Let $E_m$ be the event that $m$ adversary blocks are mined before $m$ honest blocks are mined. Using what you learnt in the lecture, present a good bound on the probability of $E_m$.

   ---

   **Answer:**

   Let $T_1^h$ be the time taken to mine the first honest block $b$, and $T_i^h$ be the inter-arrival time between the $(i-1)$-th and $i$-th honest blocks for $i > 1$. Let $T_1^a$ be the time taken to mine the first adversarial block, and $T_i^a$ be the inter-arrival time between the $(i-1)$-th and $i$-th adversarial blocks for $i > 1$. We have seen that

$T_i^h \sim$ Exponential($\lambda_h$) and $T_i^a \sim$ Exponential($\lambda_a$) for $i \geq 1$ and all those random variables are mutually independent.

Then $E_m$ is the event that

$$\sum_{i=1}^{m} T_i^h > \sum_{i=1}^{m} T_i^a. \tag{1}$$

We can then use a Chernoff bound to derive an upper bound for this probability. Define $D_i = T_i^h - T_i^a$.

$$\begin{aligned}
\Pr(E_m) &= \Pr\left(\sum_{i=1}^{m} D_i > 0\right) \\
&= \Pr\left(s\sum_{i=1}^{m} D_i > 0\right) && \text{for any } s > 0 \text{ to be determined} \\
&= \Pr\left(e^{s\sum_{i=1}^{m} D_i} > 1\right) \\
&\leq \mathbb{E}[e^{s\sum_{i=1}^{m} D_i}] && \text{using Markov's inequality} \\
&= \mathbb{E}\left[\prod_{i=1}^{m} e^{sD_i}\right] \\
&= \prod_{i=1}^{m} \mathbb{E}[e^{sD_i}] && \text{due to independence of the } D_i\text{'s} \\
&= \prod_{i=1}^{m} \mathbb{E}[e^{sT_i^h}]\,\mathbb{E}[e^{-sT_i^a}] && \text{due to independence of } T_i^h \text{ and } T_i^a \\
&= \left(\mathbb{E}[e^{sT_1^h}]\,\mathbb{E}[e^{-sT_1^a}]\right)^m && \text{due to identical distributions} \\
&= \left(\frac{\lambda_h}{\lambda_h - s}\frac{\lambda_a}{\lambda_a + s}\right)^m && \text{for any } s < \lambda_h \text{ (see below)}
\end{aligned}$$

Since this bound holds for any $0 < s < \lambda_h$, we can choose $s = \frac{\lambda_h - \lambda_a}{2}$ to minimize the right side. This gives the Chernoff bound

$$\Pr(E_m) \leq \left(\frac{4\lambda_h\lambda_a}{(\lambda_h + \lambda_a)^2}\right)^m = e^{-cm} \tag{2}$$

where $c = 2\log(\lambda_h + \lambda_a) - \log(4\lambda_h\lambda_a)$. Using part a), this can also be written as

$$\Pr(E_k) \leq (4\beta(1-\beta))^m = e^{-cm}. \tag{3}$$

where $c = 2\log(\lambda_h + \lambda_a) - \log(4\lambda_h\lambda_a) = \log\left(\frac{1}{4\beta(1-\beta)}\right)$.

Calculating $\mathbb{E}[e^{sT_1^h}]$ (can be done analogously for $\mathbb{E}[e^{-sT_1^a}]$):

$$\begin{aligned}\mathbb{E}[e^{sT_1^h}] &= \int_{t=0}^{\infty} \lambda_h e^{-\lambda_h t}\, e^{st}\, dt \\ &= \lambda_h \int_{t=0}^{\infty} e^{-(\lambda_h - s)t}\, dt \\ &= \frac{\lambda_h}{\lambda_h - s} \qquad \text{for any } s < \lambda_h.\end{aligned}$$

---

c) Let $F$ be the event that the private attack succeeds on reversing the confirmation of block $b$ under the $k$-deep confirmation rule. Express this event in terms of the events $E_m$, $m = 1, 2, \ldots$.

---

**Answer:**

Under the $k$-deep confirmation rule, the private attack can deconfirm $b$, if $m$ adversary blocks are mined before $m$ honest blocks for $m = k, k+1, k+2, \ldots$. (It cannot deconfirm for any $m < k$ because $b$ would not be confirmed yet.) Hence

$$F = \bigcup_{m=k}^{\infty} E_m. \tag{4}$$

---

d) Using your answers from previous parts or otherwise, show that the probability of $F$ decreases exponentially with $k$. (**Hint:** The union bound may be useful here.)

---

**Answer:**

Using the union bound and the Chernoff bound from part (b) above,

$$\Pr(F) = \Pr\left(\bigcup_{m=k}^{\infty} E_m\right) \leq \sum_{m=k}^{\infty} \Pr(E_m) \leq \sum_{m=k}^{\infty} e^{-cm} = \frac{1}{1 - e^{-c}}\, e^{-ck}. \tag{5}$$

Thus the probability of $F$ decreases exponentially in $k$.

---

e) Simulate the longest chain protocol under Nakamoto's private attack, and estimate the confirmation error probability for $k = 5, 10, 15, 20$ and for adversarial hash power fraction $\beta = 0.3, 0.45$. Compare your results with the analytical bound you

obtained in the previous part. (**Hint:** Make sure to repeat the runs of the protocol sufficiently often to generate reliable estimates of the error probabilities.)

---

**Answer:**

Since the error probabilities can be as low as $10^{-3}$, we should have at least $10^4$ Monte Carlo samples (better even $10^5$) to get reliable estimates.

Example Python 3 code:

```
#! /usr/bin/env python3

ks = [5, 10, 15, 20]
betas = [0.30, 0.45]

# number of Monte Carlo experiments to average over
N_monte_carlo_samples = 10000
# time horizon of the simulation: number of honest
# block creation events to simulate
N_honest_block_creation_events = 10000

import random

for beta in betas:
    for k in ks:
        print("k:", k, "beta:", beta)

        N_adversary_succeeds = 0
        for i_monte_carlo_sample in range(N_monte_carlo_samples):
            N_blocks_a = 0
            N_blocks_h = 0

            # simulate block creation events
            while N_blocks_h < N_honest_block_creation_events:
                # random chain growth
                if random.random() < beta:
                    N_blocks_a += 1
                else:
                    N_blocks_h += 1

                # the attack succeeded if the honest chain is longer than k
                # (i.e., the attacked block counts as confirmed by k-deep rule)
                # but the adversarial chain outgrows the honest chain
                if N_blocks_h >= k and N_blocks_a >= N_blocks_h:
                    N_adversary_succeeds += 1
                    break

        # output result
        print("Perror:", N_adversary_succeeds/N_monte_carlo_samples)
```

In Figure 1, the simulated probabilities are compared with the calculated bounds. The plots are shown on a log scale where a straight line corresponds to exponential decay. Both the simulated and calculated probabilities decay exponentially with $k$. For $\beta = 0.45$, the error probabilities are higher, and the decay is slower, compared to $\beta = 0.3$.

| Parameters | Upper bound on $\Pr(E_k)$ | Upper bound on $\Pr(F)$ | Simulation |
|---|---|---|---|
| $k = 5, \beta = 0.3$ | 0.4182 | 2.6138 | 0.2018 |
| $k = 10, \beta = 0.3$ | 0.1749 | 1.0931 | 0.0657 |
| $k = 15, \beta = 0.3$ | 0.0731 | 0.4572 | 0.0242 |
| $k = 20, \beta = 0.3$ | 0.0306 | 0.1912 | 0.0094 |
| $k = 5, \beta = 0.45$ | 0.9510 | 95.09 | 0.7576 |
| $k = 10, \beta = 0.45$ | 0.9044 | 90.44 | 0.6530 |
| $k = 15, \beta = 0.45$ | 0.8601 | 86.00 | 0.5892 |
| $k = 20, \beta = 0.45$ | 0.8179 | 81.79 | 0.5296 |

Table 1: Private attack confirmation error probabilities from the analytical bounds and from simulation. The bounds for both $\Pr(E_k)$ from (3), and $\Pr(F)$ from (5) are compared with the simulation output.

Since (5) only gives an upper bound on $\Pr(F)$, the simulated probabilities are always less than that upper bound (the bound may even be $> 1$). The simulated probabilities happen to be less than the upper bound for $\Pr(E_k)$ (3) as well, since most of the probability for $F$ comes from $E_k$.

However, the gap between the simulated probabilities and the upper bounds may be significant, especially when $\beta = 0.45$. The gap is because of two reasons. Firstly, the Chernoff bound only provides an upper bound, not the exact probability. Secondly, this gap can be very large when we use the union bound since the events $E_m$ are not independent.

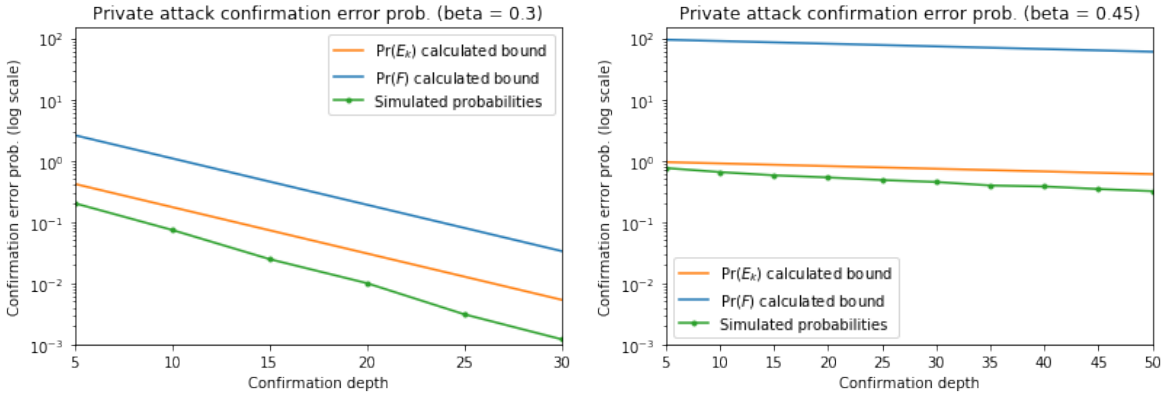



Figure 1: Private attack confirmation error - simulated probabilities compared with the calculated bounds. Note the log scale! Probabilities are shown up to $k = 30$ for $\beta = 0.3$ and $k = 50$ for $\beta = 0.45$.

2. (40%) Nakamoto's private attack was discussed in the context of reversing a confirmed block at level 1. In this problem, we will consider a more powerful attack applicable to blocks at arbitrary levels. This is an attack which is Nakamoto's private attack combined with a *pre-mining phase*. The attack is focused on reverting a transaction TX included in the block of the public chain at the $i$-th level.

   - *Pre-mining phase*: Starting from the genesis block, the attacker starts mining blocks in private to build a private chain. When the first honest block $h_1$ is mined on the genesis block, the attacker does one of two things: i) If the private chain is longer than the public chain at that moment, then the adversary continues mining on the private chain; ii) if the private chain is equal or shorter then the public chain, the attacker abandons the private chain it has been mining on and starts a new private chain on $h_1$ instead. The attacker repeats this process with all honest blocks $h_2, h_3, \ldots h_{i-1}$.
   - *Private attack phase*: After block $h_{i-1}$ is mined, the attacker will start Nakamoto's private attack from the current private chain it is working on, whether it is off $h_{i-1}$ or the one it has been working on before $h_{i-1}$ depending on which is longer.

   Answer the following questions. You may assume the same setting as in Problem 1.

   a) Suppose $\beta = 0.3$. What is the probability that the attacker will switch to $h_1$ when it is mined? What is the expected level at which the attacker is mining when $h_1$ arrives?

---

**Answer:**

For every block creation event, with probability $\beta$, it is an adversarial block, and with probability $1-\beta$ it is an honest block. The number of adversarial blocks $L_1^a$ before the first honest block arrives is thus a geometrically distributed random variable with parameter $1-\beta$, i.e. $L_1^a \sim \text{Geometric}(1-\beta)$.

The adversary switches to $h_1$ if $\{L_1^a \leq 1\}$, which happens with probability

$$\Pr(L_1^a = 0) + \Pr(L_1^a = 1) = (1-\beta) + \beta(1-\beta) = 1 - \beta^2. \tag{6}$$

The expected depth the adversary is mining at when $h_1$ arrives is

$$\mathbb{E}[L_1^a] = \frac{\beta}{1-\beta} = \frac{3}{7} \approx 0.43. \tag{7}$$

---

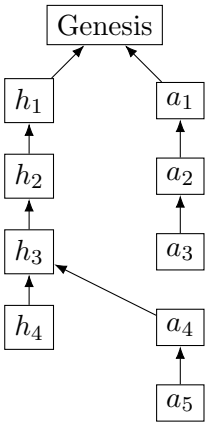
   b) Suppose honest ($h$) and adversarial blocks ($a$) are mined in the order:

$$a_1, a_2, h_1, a_3, h_2, h_3, a_4, a_5, h_4.$$

Draw the evolution of the block tree, always including both honest and adversary blocks.

---

**Answer:**




---

c) Simulate this attack for large $i$ and estimate the confirmation error probability for $k = 5, 10, 15, 20$ and adversarial hash power fraction $\beta = 0.3, 0.45$. Compare these results with those in Problem 1d). Are there significant differences? Why?

---

**Answer:**

Example Python 3 code:

```
#! /usr/bin/env python3

ks = [5, 10, 15, 20]
betas = [0.30, 0.45]

# number of Monte Carlo experiments to average over
N_monte_carlo_samples = 10000
# time horizon of the simulation: number of honest
# block creation events to simulate
N_honest_block_creation_events = 10000
# number of honest blocks during which adversary
# conducts pre-mining (= i)
N_honest_block_premining = 1000

import random

for beta in betas:
    for k in ks:
        print("k:", k, "beta:", beta)

        N_adversary_succeeds = 0
        for i_monte_carlo_sample in range(N_monte_carlo_samples):
```

```
        N_blocks_a = 0
        N_blocks_h = 0

        # simulate block creation events
        while N_blocks_h < N_honest_block_creation_events + N_honest_block_premining:
            # random chain growth
            if random.random() < beta:
                N_blocks_a += 1
            else:
                N_blocks_h += 1

                # if we are still in pre-mining phase and the honest
                # chain is longer than the adversarial chain, the adversary switches
                if N_blocks_h < N_honest_block_premining and N_blocks_h >= N_blocks_a:
                    N_blocks_a = N_blocks_h

            # the attack succeeded if the honest chain is longer than k
            # (i.e., the attacked block counts as coonfirmed by k-deep rule)
            # but the adversarial chain outgrows the honest chain
            if N_blocks_h >= k + N_honest_block_premining and N_blocks_a >= N_blocks_h:
                N_adversary_succeeds += 1
                break

    # output result
    print("Perror:", N_adversary_succeeds/N_monte_carlo_samples)
```

| Parameters | Private attack | Private attack with pre-mining ($i = 1000$) |
|---|---|---|
| $k = 5, \beta = 0.3$ | 0.2018 | 0.1962 |
| $k = 10, \beta = 0.3$ | 0.0657 | 0.0676 |
| $k = 15, \beta = 0.3$ | 0.0242 | 0.0277 |
| $k = 20, \beta = 0.3$ | 0.0094 | 0.0102 |
| $k = 5, \beta = 0.45$ | 0.7576 | 0.8679 |
| $k = 10, \beta = 0.45$ | 0.6530 | 0.8005 |
| $k = 15, \beta = 0.45$ | 0.5892 | 0.7410 |
| $k = 20, \beta = 0.45$ | 0.5296 | 0.6782 |

Table 2: Simulated confirmation error probabilities for private attack and private attack with pre-mining.

The simulation results are shown in Table 2. For $\beta = 0.3$, there is only a minor improvement. This makes sense, as the adversary cannot sustain a competitive private chain over long periods of time, pre-mining does not help much. For $\beta = 0.45$, pre-mining helps the adversary considerably, as there is now a decent chance that one of the pre-mined private chains yields a head start into the private attack. (The probability for private attack with pre-mining is smaller in one case ($k = 5, \beta = 0.3$). This would be attributed to the randomness in the simulation since the difference in the probabilities is very small.)

---

3. (30%) The chain quality of a blockchain is defined as the fraction of blocks in the public longest chain that are mined by honest nodes. Suppose that the honest mining rate is $\lambda_h$ and the adversarial mining rate is $\lambda_a$. Assume zero delay ($\Delta = 0$) so that all honest blocks will immediately be known to everyone. In the lecture, we have seen the following lower bound on the chain quality:

$$\mathsf{CQ} \geq 1 - \frac{\lambda_a}{\lambda_h} = \frac{1-2\beta}{1-\beta}, \tag{8}$$

where $\beta = \frac{\lambda_a}{\lambda_a+\lambda_h}$.

Recall the *selfish mining* strategy: The adversary always mines on the longest chain, but keeps their blocks private. Every time an honest block is mined, the adversary displaces the block by releasing one of their private blocks (if it has in the meanwhile pre-mined at least one adversarial block). Assume that when there are multiple longest chains of equal length, the adversary can choose which longest chain the honest miners will adopt. Under this assumption, the most favorable situation is that the adversary's block becomes the new longest chain, where honest miners will mine their next block. If the adversary does not have any blocks in private when an honest block is mined, it starts a new private chain on the new honest block.

a) Suppose honest ($h$) and adversarial blocks ($a$) are mined in the order:

$$a_1, h_1, a_2, a_3, h_2, h_3, h_4, a_4, h_5.$$

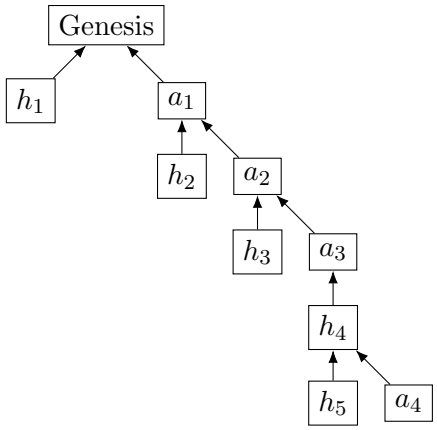
Draw the evolution of the block tree under this attack strategy, always including both honest and adversary blocks. Indicate the longest chain after all these blocks have been mined.

b) Simulate the system and plot the CQ as a function of $\beta$ to confirm that the above strategy achieves the chain quality in the lower bound, as argued in class.

---

**Answer:**

a)



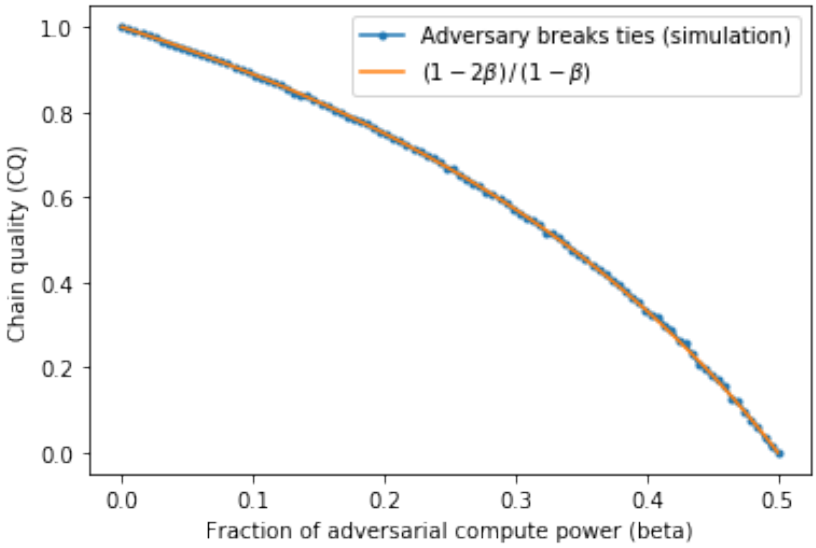

The longest chain at the end of the attack is Genesis $- a_1 - a_2 - a_3 - h_4 - a_4$. Notice how in this example, even though the adversary only mined $\frac{4}{9}$ of the total number of blocks (excluding Genesis), the adversary owns $\frac{4}{5}$ of the blocks on the longest chain!

b) Code for simulation

```
import numpy as np
import matplotlib.pyplot as plt

# Total number of blocks to simulate
num_blocks = 100000
CQ = []
betas = np.linspace(0,0.5,100, endpoint=True)
for beta in betas:
    # Number of honest and adversarial blocks in the longest chain
    h_blocks = 0
    a_blocks = 0
    # Current length of the adversary's private chain
    priv_len = 0
    for i in range(num_blocks):
        # When an adversarial block arrives, add it to the private chain
        if np.random.rand() < beta:
            priv_len += 1
        else:
        # When an honest block arrives, replace it with an adversarial block
        # if the adversary has a private chain
            if priv_len > 0:
                a_blocks += 1
                priv_len -= 1
        # Otherwise, add the honest block to the longest chain
            else:
                h_blocks += 1
    CQ.append(h_blocks / (h_blocks+a_blocks))
```

The figure below shows the simulated chain quality as a function of $\beta$, which matches the calculated limit $\frac{1-2\beta}{1-\beta}$.


Adversary breaks ties (simulation)
$(1-2\beta)/(1-\beta)$
Chain quality (CQ)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.1
0.2
0.3
0.4
0.5
Fraction of adversarial compute power (beta)