Double Selfish Mining Attack

B.S.S.R. Saran **K. Sree Nikhil** **V. Mahanth Naidu**

CS 765 Assignment - 2

1 Introduction

In this report, we present a simulation of a double selfish-mining attack on top of the discrete-event simulator for a P2P cryptocurrency network. In this scenario, there are two such individual selfish miners who do not collaborate. Each behaves as if he is the only selfish miner in the system, unaware of the presence of the other attacker.

- **Honest miner:** Each honest node mines on the longest chain visible to it. Tie breaks are resolved according to bitcoin rules, that is in case it sees multiple such longest chains, it mines on the first of these which it sees.
- **Selfish miner:** Each selfish miner considers the longest visible chain (LVC) to him (excluding his current private blocks) as if it were the honest chain, and tries to mine selfishly. Note that the LVC may contain blocks released by the other selfish miner, but this selfish miner thinks that all other miners are honest.

2 Implementation Details

The attack begins with the selfish miner trying to generate a secret chain on the genesis block. The rules for releasing secret blocks or starting a new attack on another block (that is, creating a new secret chain starting from another publicly visible block) are described below:

- If the lead of the selfish miner is 1 block over the LVC, and the lead now becomes zero, then the selfish miner immediately broadcasts the block he has mined secretly. The selfish miner continues to mine on top of his own block. If the LVC increases further in length, the selfish miner starts a new attack, starting from the last block of the LVC. However, if instead the selfish miner mines a block then he releases it immediately and starts a new attack starting from the block that he just released.
- If the lead of the selfish miner over the LVC is 2 blocks and then one block gets added to the LVC, then the selfish miner broadcasts immediately all

the blocks he has mined secretly. He starts a new attack on the last block of the longest chain visible to him.

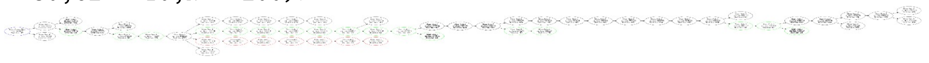
- If the lead of the selfish miner over the LVC is greater than 2, as soon as the LVC increases in length by 1 block, then the selfish miner makes public one more block, releasing a subchain that ends with that block which enters into competition with the new block at the end of the LVC. The selfish miner keeps mining on top of his secret chain.

- At any point in time, if the LVC exceeds the length of the selfish miner's private chain, then he starts a new attack on the last block of the longest chain visible to him. The selfish miner does not forward blocks generated by other nodes. This is so that his own blocks propagate faster than blocks generated by others.

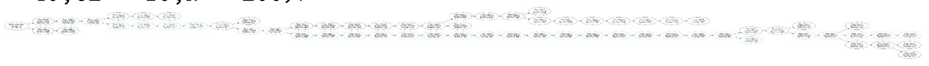
3 Diagrams

Note: The input parameters were total number of events`(N)` in the simulation, number of nodes in the network`(n)`, mean times for the exponential distributions`(Ttx and Tk)`, hashing power of attacker 1`(C1)` and hashing power of attacker 2`(C2)`.Blocks generated by attacker 1, attacker 2, and honest miners are in green, red, and black color respectively.

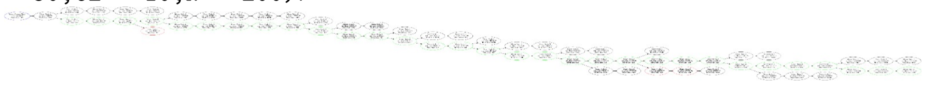
- Blockchain Tree for the input paramaters `(n = 10,Ttx = 500,Tk = 600,C1 = 30,C2 = 10,N = 200)`.



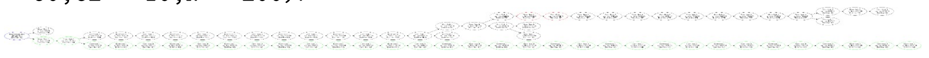
- Blockchain Tree for the input paramaters `(n = 10,Ttx = 500,Tk = 600,C1 = 40,C2 = 10,N = 200)`.



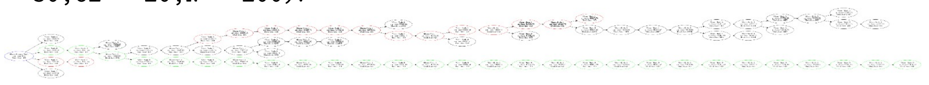
- Blockchain Tree for the input paramaters `(n = 10,Ttx = 500,Tk = 600,C1 = 50,C2 = 10,N = 200)`.

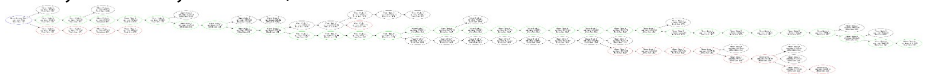


- Blockchain Tree for the input paramaters `(n = 10,Ttx = 500,Tk = 600,C1 = 60,C2 = 10,N = 200)`.



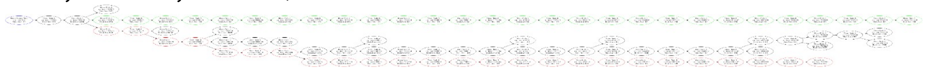
- Blockchain Tree for the input paramaters `(n = 10,Ttx = 500,Tk = 600,C1 = 30,C2 = 20,N = 200)`.



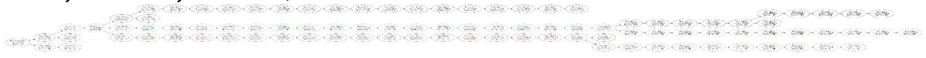
- Blockchain Tree for the input paramaters (`n = 10,Ttx = 500,Tk = 600,C1 = 40,C2 = 20,N = 200`).


- Blockchain Tree for the input paramaters (`n = 10,Ttx = 500,Tk = 600,C1 = 50,C2 = 20,N = 200`).

- Blockchain Tree for the input paramaters (`n = 10,Ttx = 500,Tk = 600,C1 = 60,C2 = 20,N = 200`).

- Blockchain Tree for the input paramaters (`n = 10,Ttx = 500,Tk = 600,C1 = 30,C2 = 30,N = 200`).


- Blockchain Tree for the input paramaters (`n = 10,Ttx = 500,Tk = 600,C1 = 40,C2 = 30,N = 200`).

- Blockchain Tree for the input paramaters (`n = 10,Ttx = 500,Tk = 600,C1 = 50,C2 = 30,N = 200`).


- Blockchain Tree for the input paramaters (`n = 10,Ttx = 500,Tk = 600,C1 = 60,C2 = 30,N = 200`).

4 Graphs

- For analysis, we fixed all other parameters except hashing power of the attacker 1(`C1`) and hashing power of the attacker 2(`C2`). So the graphs were generated by taking `n = 10, Ttx = 500, Tk = 600, N = 200`.
- The Blue line indicates MPU for attacker1, the Orange line indicates MPU for attacker2, and the Green line indicates MPU for overall nodes. X-axis contains the hashing power of attacker1(`C1`) and Y-axis contains the MPU ratios.

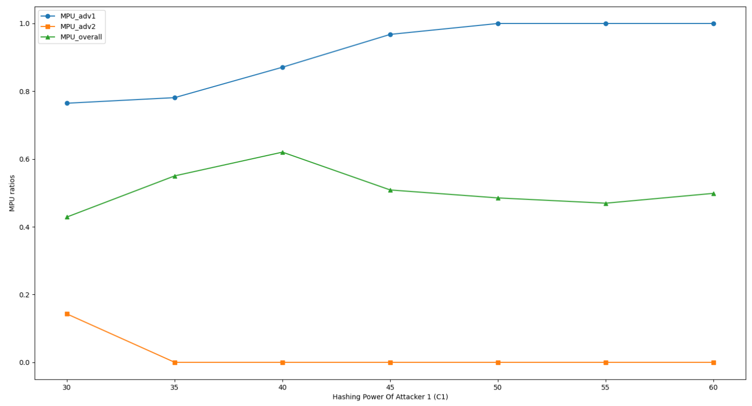


Figure 1: Hashing Power Of Attacker 2(C2) is 10%

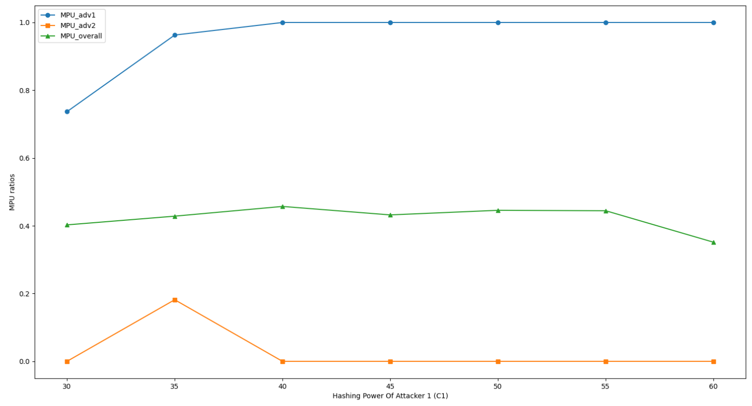


Figure 2: Hashing Power Of Attacker 2(C2) is 20%

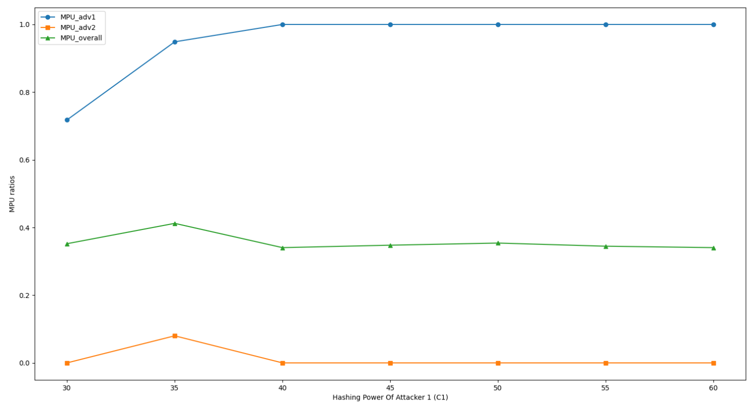


Figure 3: Hashing Power Of Attacker 2(`C2`) is 30%

5 Analysis

C1	C2	$\mathbf{MPU_{adv1}}$	$\mathbf{MPU_{adv2}}$	$\mathbf{MPU_{overall}}$
30	10	0.76470588	0.14285714	0.42894736
40	10	0.87096774	0.00000000	0.62030303
50	10	1.00000000	0.00000000	0.48529411
60	10	1.00000000	0.00000000	0.49871428
30	20	0.73684210	0.00000000	0.40277777
40	20	1.00000000	0.00000000	0.45714285
50	20	1.00000000	0.00000000	0.44578313
60	20	1.00000000	0.00000000	0.35164835
30	30	0.71794871	0.00000000	0.35227272
40	30	1.00000000	0.00000000	0.34065934
50	30	1.00000000	0.00000000	0.35416666
60	30	1.00000000	0.00000000	0.34065934

- If either the hashing power of attacker1 or attacker2 is zero then the number of blocks generated by the respective miners will be zero implying that while calculating the MPU ratio for that node or miner both the numerator and denominator will be zero. Then this case will become a selfish mining attack with only one attacker.
- In our analysis, we initially kept the hashing power of attacker2 (`C2`) constant while increasing the hashing power of attacker1 (`C1`). Subsequently, we varied the hashing power of attacker2 (`C2`) and generated graphs by incrementing the hashing power of attacker1 (`C1`) for each value of `C2`.

- Since the hashing power of attacker1 and attacker2 combined is `C1+C2`. The remaining `100-C1-C2` will be distributed equally among the remaining `n-2` honest miners which implies that the hashing power of each honest miner is much less when compared to that of C1 or C2 for most of the cases taken. Hence, most of the blocks in the longest chain will be either from attacker1 or attacker2 depending on C1 and C2.

- If the hashing power of attacker1 `(C1)` and hashing power of attacker2 `(C2)` are nearly the same, then the longest chain generally contains blocks from both the attackers which shows that both the attackers lose their secret chain.

- If the hashing power of attacker1 `(C1)` is greater than the hashing power of attacker2 `(C2)` then the longest chain contains most of the blocks generated by attacker1. This is because attacker1 generates blocks faster than others and due to the slow rate of attacker2 he has to lose his secret chain to attacker1 frequently and vice-versa.

- For a fixed value of `C2`, on increasing `C1` the MPU of attacker1 keeps on increasing and reaches an optimum value equal to 1 implying that all the blocks generated by attacker1 will be in the longest chain. As `C1` increases, the MPU of attacker2 decreases and finally becomes 0 implying that the longest chain doesn't contain blocks generated by attacker2 and vice-versa.

- If the hashing power of attacker2`(C2)` is zero and if the hashing power of attacker1`(C1)` is less than 50, then `100-C1-C2` will be greater than 50. In this case, as we increase `(C1)` the MPU overall decreases and reaches an optimum equal to 0.5 i.e., when `C1` equals 50. If we further increase `C1` after 50, the MPU overall will be 0.5 because the attacker won't release his secret chain until the honest miners release their blocks. The same pattern occurs if we interchange `C1` and `C2`.

- If both the hashing power of attacker1`(C1)`, attacker2`(C2)` are non-zero and `C1` is large such that both the `C2` and `100-C1-C2` are nearly same. In this case, as we increase `C1` then the MPU overall decreases and reaches an optimum value equal to 0.33 when both the hashing power of attackers equals the total hashing power of the honest miners. After this, if we increase `C1`, the MPU overall will be nearly 0.33 because the attackers won't release their secret chains until the honest miners release their blocks.

- If the attackers possess significantly lower hashing power compared to honest miners, it is more likely for multiple branches to emerge (forking becomes highly probable) due to the reduced likelihood of a successful selfish mining attack. Consequently, the MPU of each attacker, as well as the overall MPU, decreases.

- From the above three graphs, for most of the cases as the hashing power of the attacker increases we can clearly see that the MPU overall lies between 0.3 to 0.6.