Homework 2

Simulation of a P2P Cryptocurrency Network

Chaitanya Garg (210050039), Omm Agrawal (210050110), Pulkit Goyal (210050126)

31st March 2025

1 Running Instructions

```
$ python3 main.py --help
usage: main.py [-h] -n NUM_PEERS -m RATIO_MALICIOUS -o TIMEOUT
               -t TRANSACTION_INTERARRIVAL -b BLOCK_INTERARRIVAL -s
              SIM_TIME [-f FOLDER] [-r] [-c]

Process CLI Inputs.

optional arguments:
  -h, --help            show this help message and exit
  -n NUM_PEERS, --num_peers NUM_PEERS
                        Total Number of Peers
  -m RATIO_MALICIOUS, --ratio_malicious RATIO_MALICIOUS
                        Fraction of Malicious Peers
  -o TIMEOUT, --timeout TIMEOUT
                        Timeout Time (seconds)
  -t TRANSACTION_INTERARRIVAL, --transaction_interarrival TRANSACTION_INTERARRIVAL
                        Mean Interarrival Time for Transaction Generation (seconds)
  -b BLOCK_INTERARRIVAL, --block_interarrival BLOCK_INTERARRIVAL
                        Mean Interarrival Time of Blocks (seconds)
  -s SIM_TIME, --sim_time SIM_TIME
                        Simulation Time (seconds)
  -f FOLDER, --folder FOLDER
                        Folder to store results
  -r, --remove_eclipse  Remove Eclipse Attack from Malicous Nodes (only selfish mining)
  -c, --counter_measure
                        Add counter measure in Honest Nodes against eclipse attack.
```

All arguments mentioned in the Usage Instructions are required, except `-f, -folder` flag. The default folder name is created using all the other parameters.

1.1 Network structure

As per the requirements of the problem statement, we have a public P2P network and private overlay network consisting only of malicious nodes.
The below graph shows both the networks for a total of 50 peers and 30% malicious peers

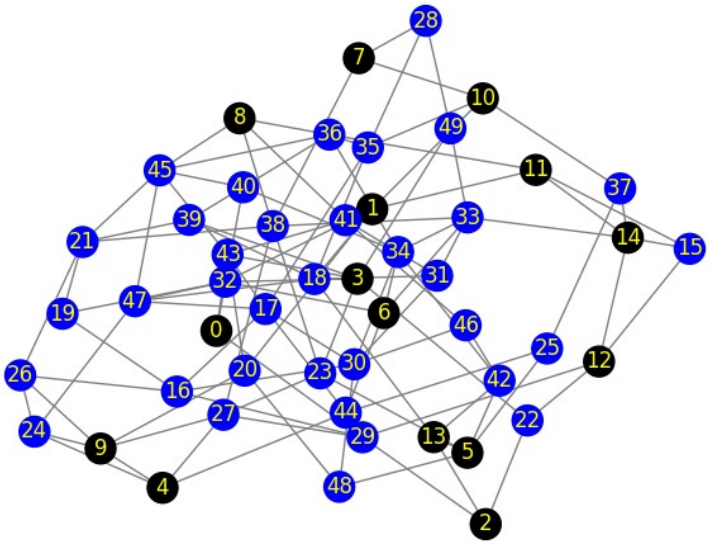


Figure 1: Public blockchain network connecting both honest and malicious nodes

In the figure below **Node 0** is the ringmaster (chosen randomly among all malicious)

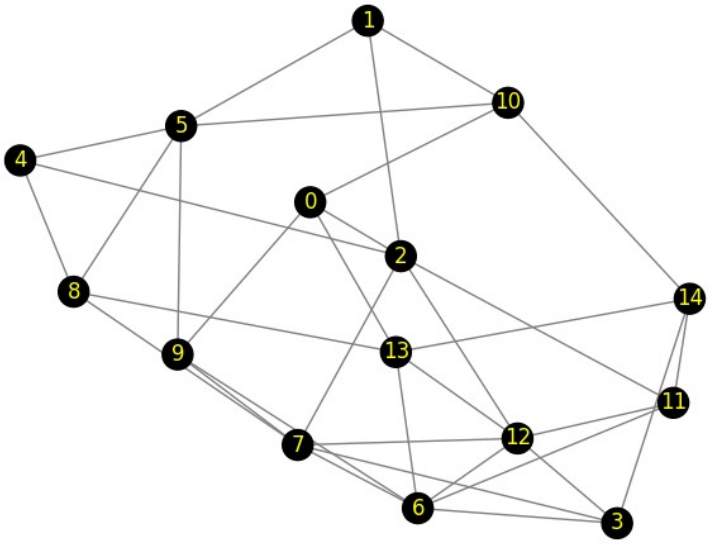


Figure 2: Private Overlay network connecting malicious nodes for their effective communication

2 Analysis of various metrics

First we conduct the analysis of ratios by keeping the timeout time constant. The timeout time is chosen to be 1s since the worst case back and forth communication between 2 peers is slightly more than 1s.

2.1 Contribution of malicious nodes in the longest chain in blockchain tree

The following parameters were kept constant throughout :

Timeout time:	1 second
Interarrival Time of Transactions:	0.2 seconds
Interarrival Time of Blocks :	10 seconds
Simulation time:	1000 seconds

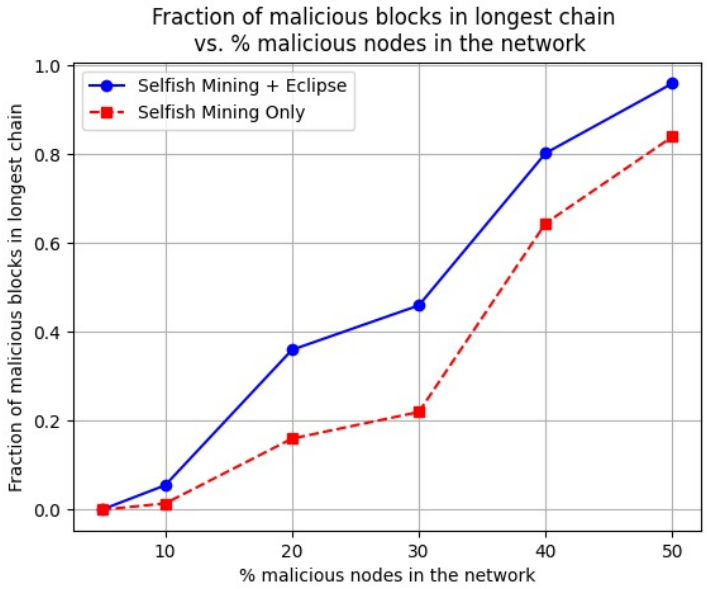


Figure 3: Ratio showcasing the contribution on malicious nodes in the longest chain

Following are the observations and explanations

- On increase in the percentage of malicious nodes, the percentage of blocks in longest chain for malicious nodes increases as expected since their hashing power increases.
- A more interesting observation is that somewhere between 30-40 % of hashing power, the % contribution crosses the y=x line as we studied in class.
- Also we can see that eclipse attack improves the selfish mining efficiently in all cases. This happens because honest nodes are prevented from knowing the actual longest chain, causing wastage of hashing power in creating more forks.

2.2 Fraction of malicious blocks that land up in longest chain

This metric would show the efficiency of the attack i.e. are the malicious blocks really landing up in longest chain

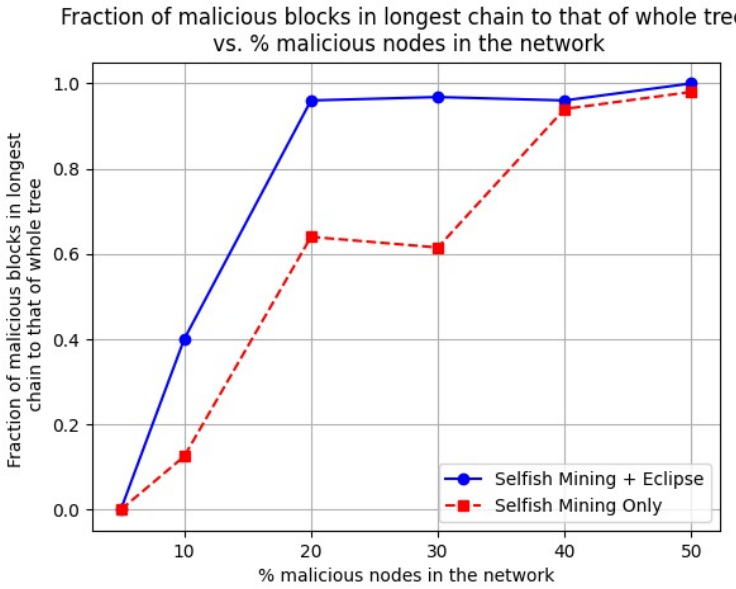


Figure 4: Ratio showcasing that the relation between the attack and longest chain efficiency

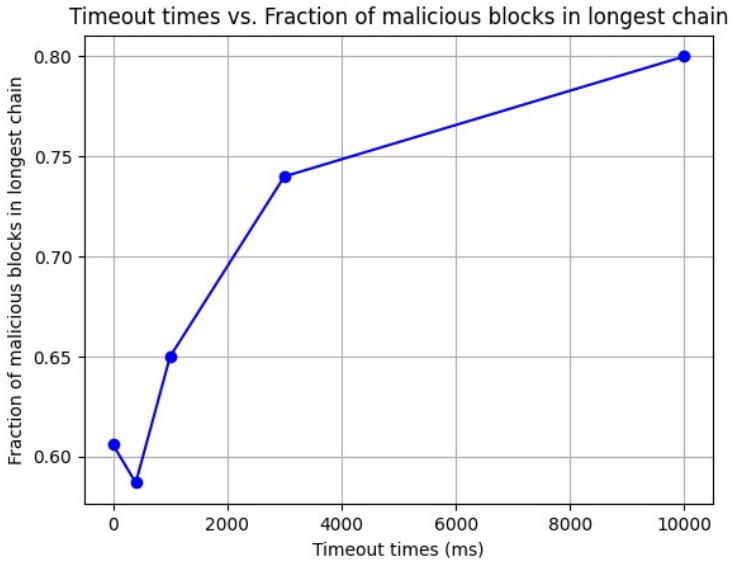
Following are the observations and explanations

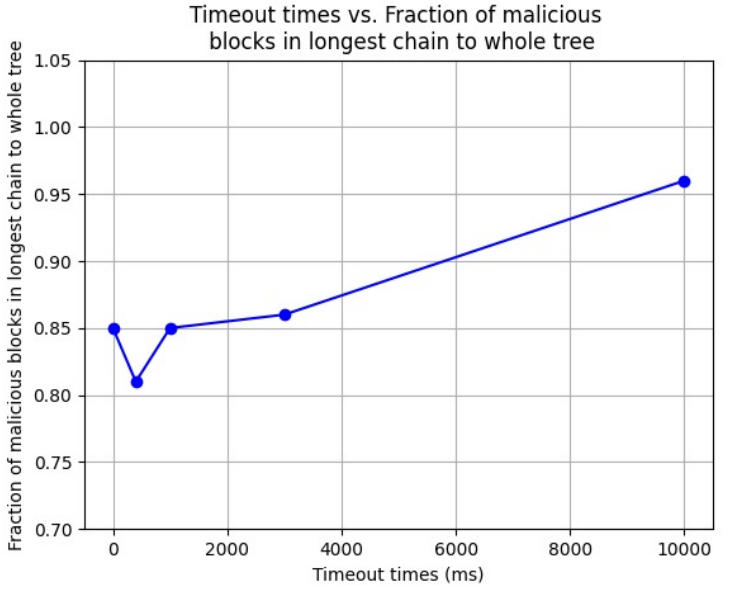
- As expected, the above ratio increases with the percentage increase in malicious nodes, this is because as their hashing power increases, there is a better chance of them to take a lead in selfish mining and then broadcast.
- Also we see that eclipse attacks helps in quickly saturating this ratio to 1. Since eclipse attack prevents the truth of the blockchain tree to reach honest miners, there is an increased probability that malicious can improve the longest chain with its ground truth knowledge in the time honest know about the longest chain.

3 Impact of different timeout times

We calculated the above 2 ratios by varying the timeout times. The below graph shows the results. Following parameters were used :

Malicious peers:	30%
Interarrival Time of Transactions:	0.2 seconds
Interarrival Time of Blocks :	10 seconds
Simulation time:	1000 seconds





Observations :

- On increasing the timeout time, the eclipsing impact increases and hence malicious peers are able to get a lead on the longest chain by branching the hashing power of the honest peers
- With increasing eclipsing impact, the malicious nodes are also able to keep honest nodes waiting longer for honest blocks, but getting honest blocks quickly. This enables them to strategically release private chain during these delays to reduce orphaning of malicious blocks, improving the percent of malicious blocks that end up in the longest chain.

3.1 How does increasing the timeout time (Tt) affect block propagation in the presence of an eclipse attack

The timeout time is a direct indicator of the strength of eclipse attack. As the eclipse attack becomes stronger, the block propagation in the network (the public network) becomes slower.

But how to gauge this? Although honest peers will receive all the block info eventually (unless a peer is completely surrounded by malicious peers which is an unlikely scenario), the slower block propagation in the network will lead to a higher fork number in the blockchain tree since different peers will continue working on the same block without the information that someone has already created a block.

Also to see the effect of eclipse attack we should only consider the **forks between honest peers on an honest block**. Let's call them effective forks for the discussion.
We show this by the different block chain tree images below :
The following parameters were used

Malicious percentage:	30%
Interarrival Time of Transactions:	0.2 seconds
Interarrival Time of Blocks :	10 seconds
Simulation time:	1000 seconds

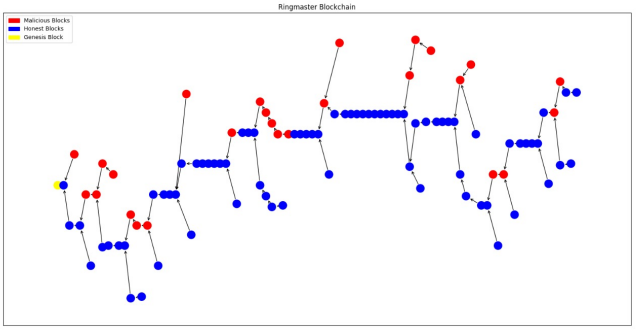


Figure 5: No eclipse attack - we see only see 3 effective forks

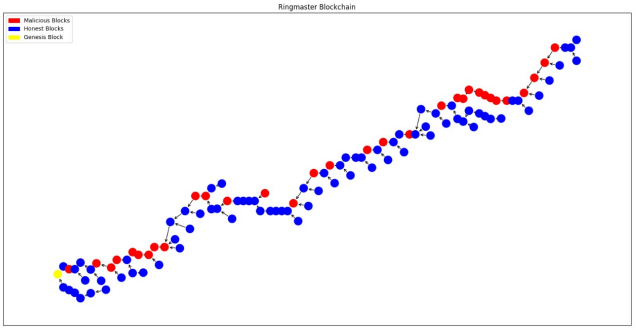


Figure 6: 1 second timeout - we see only see 8 effective forks

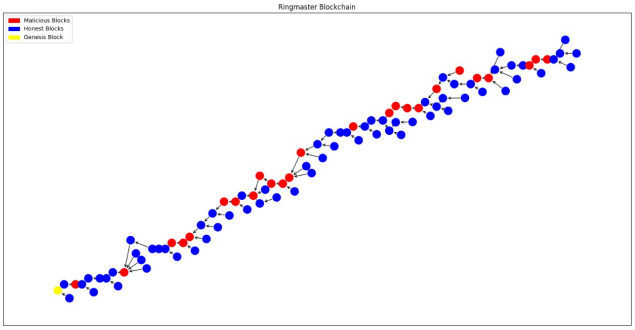


Figure 7: 10 seconds timeout - we see only see 13 effective forks with multi fork at some blocks

3.2 Impact of malicious peer percentage on eclipsing

We can also see that not only timeout but the percentage of malicious nodes can impact the eclipsing extent. We already saw the eclipsing for 30%. Let's see for 50% with timeout of 1 second

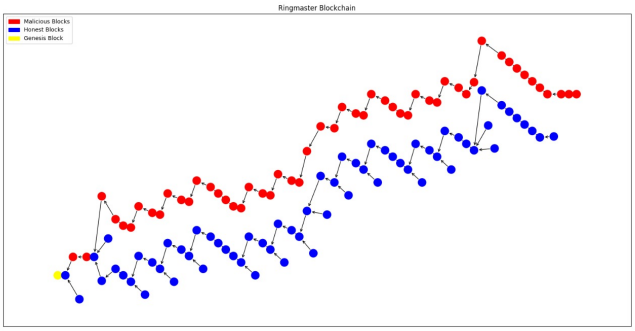


Figure 8: Significant increase in **effective** forks

We can also see that the malicious are the ones completely dominating the longest chain as expected.

4 Countermeasure Against Eclipse Attack

4.1 Overview of Threat

The two-step block propagation protocol (hash broadcast followed by full block retrieval upon request) creates a vulnerability: if an honest node sends a "get" request to a malicious peer, it must wait for a timeout before retrying with another peer. This delay can be exploited by malicious nodes to affect the timely dissemination of honest blocks. The protocol suggests waiting for timeout before sending "get" request for same hash again to reduce network overheads, but that also makes this attack possible. Our aim is to reduce this bottleneck while ensuring minimal additional strain on the network.

4.2 Countermeasure - Dynamic Trust Management

To mitigate the impact of such attacks, the following mechanism is introduced for each honest miner:

Trust List Maintenance: For every connection (peer), the honest node maintains a list of block hashes for which a "get" request has been issued but the corresponding full block has not been received. This list acts as an indicator of the peer's reliability.

Dynamic Trust-Based Selection:

- A connection is deemed not trustworthy if it has at least one pending block request that has timed out.
- If the connection sends the block even after time out, it becomes trusted again. This is to ensure false positives where honest connections sent the block but delay was high don't last very long.
- When the node receives the same block hash from another peer, if the initial connection is marked as untrustworthy due to a timeout, the node immediately sends a "get" request to this new peer without waiting for the timeout.
- When choosing a connection for subsequent "get" requests (after a timeout occurs), the node does not rely solely on a first-in-first-out (FIFO) order. Instead, it preferentially selects peers that are deemed trustworthy based on historical responsiveness.

4.3 Evaluation of the Countermeasure

Figure 9 clearly show that countermeasure suggested decreases the Contribution of malicious blocks in the longest chain by as much, as Eclipse Attack increases over selfish Mining. This shows that our suggested countermeasure is very effective in mitigating Eclipse Attack.

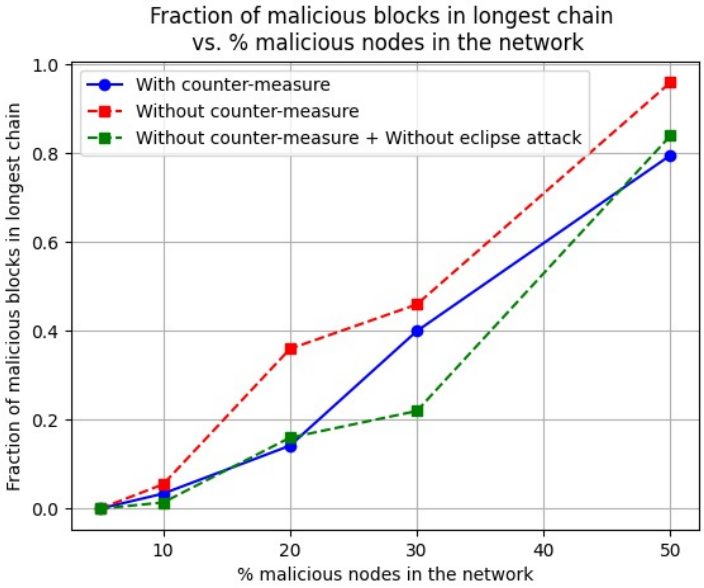


Figure 9: Figure showcasing the effect of attack countermeasure technique

4.4 Over Potential Improvements

It is quite possible for malicious nodes to enhance the Eclipse attack to work, to some extent, even when honest nodes employ this countermeasure. One such idea can be as follows:

- As Malicious Node, when we want to send a new block hash to an honest connection and want the honest peer to wait for time out on us, we first clear all outstanding get requests by this peer, by sending the previously requested full blocks. Then send the current hash.

- Clearing outstanding requests makes honest peer consider us trust worthy again, and would wait for time out on us on the next get request.
- Honest miners had a mechanism to clear false positives, and here we exploit that mechanism.

The Countermeasure can be made more resilient towards these variations of eclipse attack by:

- **Statistical Reputation System:** Instead of a binary "trustworthy" or "not trustworthy" label, implementing a scoring or reputation system for each peer could provide a more granular measure of reliability. Metrics such as average response time, success rate of full block delivery, and historical performance could be combined to form a dynamic reputation score.
- **Adaptive Timeout Mechanism:** Introduce adaptive timeout intervals based on network conditions and historical peer performance. A shorter timeout may be more effective during periods of high network congestion or when malicious activity is suspected.
- **Periodic Reassessment:** Periodically reassessing and updating the trust metrics can help mitigate issues if a previously unresponsive node improves its performance or vice versa. This ensures that the trust list reflects the current state of the network.