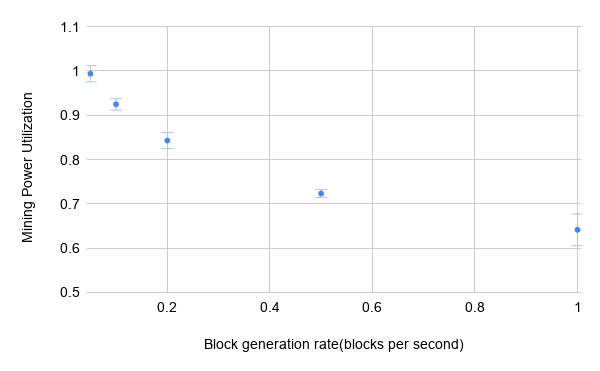
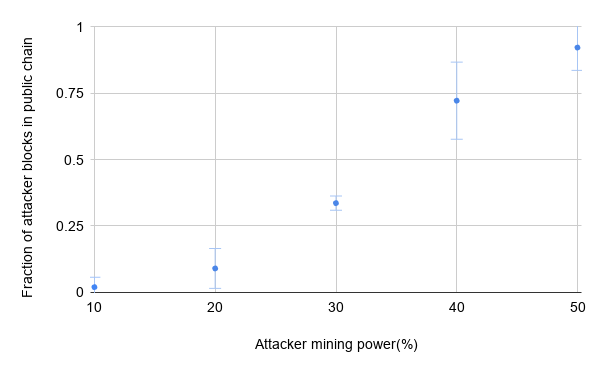
**Graph 1:** **Mining Power Utilization vs Block Generation Rate**

We can see that Mining Power Utilization is at the highest when Block Generation rate is as low as 0.05 blocks per second. And gradually decreases as Block Generation rate increases. This is due to higher chance of forking at higher block generation rate.

**Graph 2: Fraction of attacker blocks in public chain vs Attacker mining power**



Fraction of attacker blocks in public chain denotes the fraction of total revenue earned by the attackers. In case of a tie of honest chain and attacker chain, gamma denotes the fraction of honest miners that adopt the attacker chain. In our case, we assume gamma is 0 due to low propagation delay, i.e. all honest miners adopt honest chain as they get it before attacker chain.

Therefore, if attacker mining power is > 33%, they will earn a higher revenue than their share.

This can be seen in graph, low attacker mining power(10%,20%) earns them less revenue than their share.And higher attacker mining power(40%,50%) earns them much higher revenue than their share. An attacker mining power of 30% gives them the same or slightly higher share of revenue.

**Longest Chain determination:**

In our client code, we have used sqlite3 database to store all the incoming blocks.

Each block received also contains the height attribute which signifies at which level of blockchain the block belongs to.

Along with it we have used two data structures to maintain hash of blocks:

(i) local\_chain

(ii) dictionary\_of\_hash

The dictionary of hash is a dictionary that stores hash of current block and hash of the previous block as key value pairs.

For example previous hash of genesis = '0'

So, dictionary\_of\_hash['9e1c'] = '0'

Let's suppose next block comes pointing to genesis block,

we have dictionary\_of\_hash['7eff'] = '9e1c'

and it goes on likewise.

The local chain contains the hash of blocks in the longest chain received till now as it only appends the incoming blocks if the length of the chain is less than the block height. If block height is greater than its length, it automatically appends the hash of that block followed by changes in its previous hashes(if previous hash of the block is not the same as the last hash value of the local chain) referencing dictionary\_of\_hash for the said purpose. In case a

block comes with height less than the chain length its hash value is not added to local\_chain. Thus it maintains the longest chain of hashes locally. The height of the sent block (if generated by miner) is also determined by the length of the local chain(height = 1+ length(local\_chain))