Axel BATTUT

ESILV

CryptoFinance

Selfish Mining Simulation :

How it works :

The selfish mining strategy is a tactic used by miners in a proof-of-work blockchain system, in our example, Bitcoin, to increase their chances of earning 'coinbase' rewards. The strategy is based on the observation that if a miner finds a new block, they can temporarily hide it from the rest of the network, and continue to mine on top of it. This gives the miner an advantage over other miners, as they are now working on a longer chain, and therefore have a higher probability of solving the next block and earning the reward.

By withholding a block, the miner can also prevent other miners from working on the same chain, potentially slowing down the rate at which new blocks are added to the network. This can lead to a situation where the miner who is using the selfish mining strategy has a significant portion of the mining power on the network.

The selfish mining strategy can be used to earn more coinbase rewards, but it also can cause negative consequences on the blockchain network, such as slowing down the rate at which new blocks are added to the chain, and increasing the risk of a chain fork. As much as it can be considered unethical because it can decrease the decentralization of the network, we aim at studying different scenarios and evaluate at which point the selfish mining strategy can be profitable to use over the conventional mining.

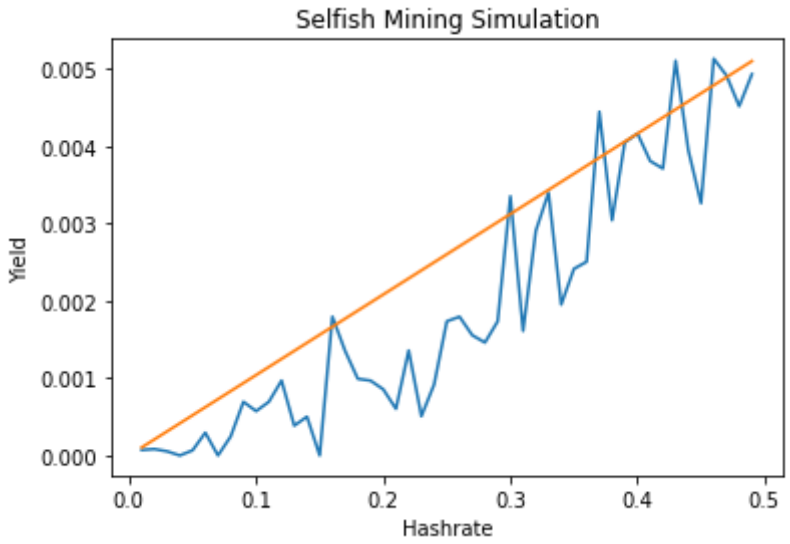
Code Review :

First, the function attackcycle() simulates an attack on a blockchain using a given hashrate (q) and connectivity (connectivite) on an existing blockchain (chaine) for a given number of cycles (NbrCycles). The function creates a new blockchain (bchainAttaquant) and modifies it based on the existing blockchain. The attacker mines blocks with a probability of q and the function simulates different scenarios based on the number of blocks mined by the attacker and the official blockchain. The function also simulates a difficulty adjustment mechanism, where the mining time is adjusted based on the number of blocks mined before the adjustment. The function also keeps track of the time it takes to mine the blocks, the number of blocks mined by the attacker and the official blockchain, and the expected earnings for the attacker. At the end, the function returns the expected gain for the attacker.

Then the function calculateRatiosForListOfHashrates() takes in two parameters, gamma and NbrCycles. It creates two lists of length 49, listeDeHashrate and listeEsperanceGains, initially filled with zeroes. The function then loops through the range 1 to 50, and for each iteration, it creates a new instance of the Blockchain class, calls the attackCycle function with the current iteration value divided by 100 as the q parameter, the value of gamma as the connectivite parameter, the newly created Blockchain instance as the chaine parameter, and the value of NbrCycles as the final parameter. The returned value from the attackCycle function is then stored in the

listeEsperanceGains at the current index minus 1. The value of the current iteration divided by 100 is then stored in the listeDeHashrate at the current index minus 1. Finally, the function returns both lists.

Results :



The simulation of the selfish mining strategy versus conventional mining shows that for a relative hashrate of greater than 42%, the attacker would benefit from using the non-conventional method, which involves privately mining and then overwriting the official blockchain when possible. Specifically, the threshold at which this method becomes profitable is reached at exactly q = sqrt(2) – 1.

1 + 2 Simulation :

How it works :

The 1+2 mining strategy attack is a type of selfish mining attack on the Bitcoin blockchain. It involves a miner or a group of miners. The miner will begin by mining blocks privately and not broadcasting them to the network. Once the miner has mined two blocks, they will broadcast the first block to the network and extend the second block privately. This allows the miner to have a lead of two blocks over the rest of the network, which increases their chances of finding the next block and earning the block reward. In this scenario, we are trying to find out if a specific hashrate makes the 1+2 mining more profitable than the honest mining.

Before reviewing the code, we need to first list all possible block scenarios in this mining situation.

w	R	H	Pb
B	0	1	p
AAA	3	3	q^3
AAB	2	2	pq^2
ABA	2	2	pq^2
ABB	0	2	p^2q

The above table shows different conditions of mining strategies, where "w" is the number of blocks mined, "R" is the number of blocks revealed, "H" is the number of blocks held by the miner, and "Pb" is the probability of the event occuring. The rows represent different scenarios.

The first row represents the conventional mining strategy, where the miner mines and immediately publishes the block, resulting in "w" = 0, "R" = 1 and "H" = 0. The probability of this event happening is "p".

The second row represents the selfish mining strategy, where the miner mines and holds the block, while continuing to mine the next one. When the miner finds another block, they reveal the one they have been holding and the probability of this event happening is "q^3"

The third and fourth rows represent the scenario where both the attacker and the honest miner find a block at the same time. The selfish miner reveals one of the blocks they have mined and holds the other. The probability of this event happening is "pq^2"

The fifth row represents the scenario where the attacker has mined more blocks than the honest miner. In this case, the attacker reveals all the blocks they have mined and the probability of this event happening is "p^2q"

Code Review :

The main function attack() takes in two arguments, attackerHashrate and numberAttack, and simulates the mining process for numberAttack cycles.

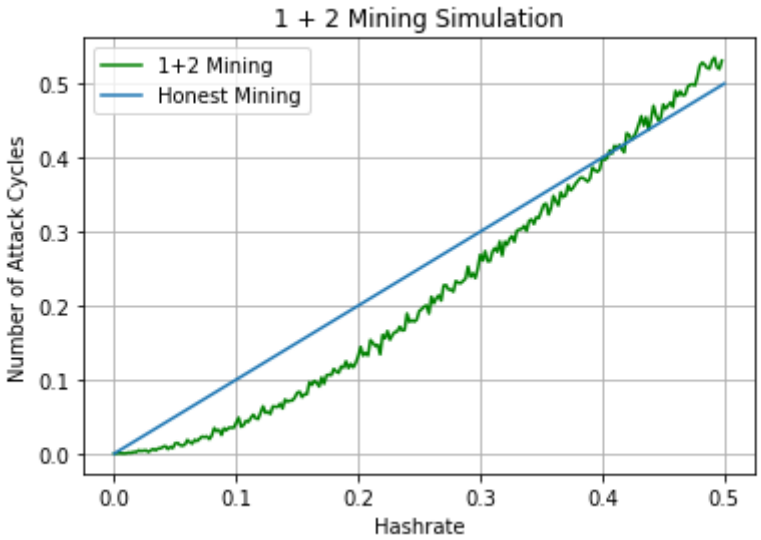
This function creates three lists which are used to store the gain, height, and difficulty of the mining process for each attack cycle.

The function attacks() takes two arguments, cycles and q, and simulates the mining process for cycles number of cycles, where q is the probability that the attacker mines a block. It returns a list of tuples, where each tuple contains the number of blocks mined by the attacker and the total height of the blockchain.

The function goodCycles() is used to get the number of attack cycles from the user and ensures that the input is a number between 0 and 10000.

The main() function takes the input of the attacker's hashrate, number of attack cycles from the user, and calls the functions attack and attacks and plots the results of the simulation.

Results :



It is observed that for a certain hashrate (41.5%), the 1 + 2 attack is more profitable than honest mining.

Optimal Mining :

How it works :

The goal of this code is to determine the point at which it becomes more profitable for a miner to switch from traditional mining to an alternative strategy. To do this, it calculates the expected maximum gain for a miner as a function of their hash power. By plotting this relationship and analyzing the results, the code aims to identify the threshold at which the alternative strategy becomes more profitable.

Code Review :

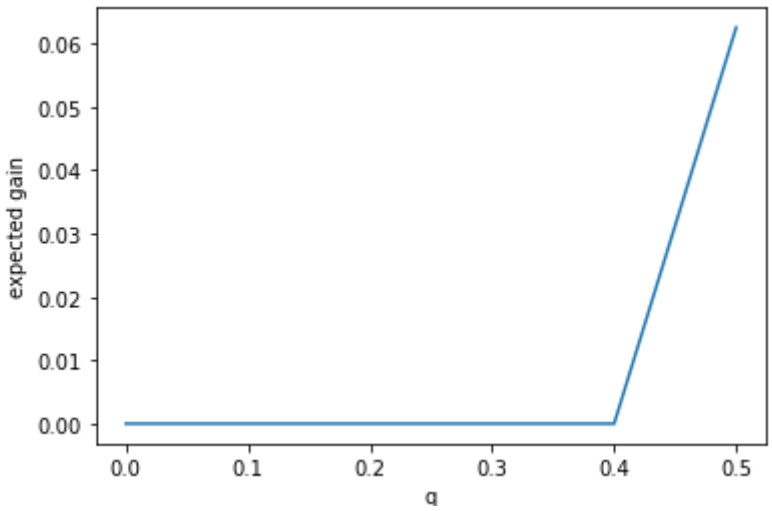
The first function, Max(a,b) is a simple utility function that returns the maximum of the two inputs, a and b.

The second function, E(a,h,n,q,c) is the main function that calculates the expected gain of the individual in the game-theoretic model. It takes in 5 parameters: a, h, n, q, and c. The function uses recursion to calculate the expected gain for different values of a, h, n, q, and c. The function has three different branches, one for each of the three conditions (a > h, a == h+1, a <= h)

The function simulation_curves() creates a plot of the expected gain for different values of q. It calls the E(a,h,n,q,c) function for different values of q and stores the results in a list. It then plots the results using matplotlib library.

The last lines of the code call the simulation_curves() function to create the plot and print some outputs.

Results :



From the results of the code, it can be seen that for values of q greater than 42%, it is more profitable for the miner to use the non-conventional mining strategy and abandon the honest mining strategy.

Conclusion

In conclusion, the article discussed three different mining strategies: the optimal mining strategy, the selfish mining strategy, and the 1+2 mining strategy.

These three strategies have been studied to understand the behavior of miners in a decentralized network and to identify potential issues that may arise in the mining process. By understanding the trade-offs between these strategies, it is possible to identify the conditions under which one strategy is more profitable than the others. Additionally, these strategies can help to inform the design of consensus mechanisms to mitigate the negative effects of selfish mining.

In summary, the study of these mining strategies is important for understanding the behavior of miners in a decentralized network, identifying potential issues that may arise in the mining process, and informing the design of consensus mechanisms.