Detecting Selfish Mining in Bitcoin

Fangyang Cui

12/7/2015

Selfish mining is an attack on the Bitcoin network discovered by Ittay Eyal and Emin Gün Sirer[1]. The attack involves a miner selectively choosing when to publish blocks that they have discovered instead of doing so immediately. The purpose of the attack is to obtain a two block or longer lead over the rest of the Bitcoin network. The selfish miner only publishes their blocks when the network has caught up to the point where the honest miners are one block behind the selfish miner. In their paper, Eyal and Sirer show that a selfish miner following these instructions is able to obtain a higher proportion of the Bitcoin blocks than they deserve based on their hashrate.

Miners performing selfish mining attacks can produce undesirable consequences for the rest of the network. Because a successful selfish mining attack occurs when the selfish miner releases a set of blocks that form a longer chain than the chain created by the rest of the network, many orphan blocks will be created. Whenever there is an orphan block, there is a potential for negative consequences, so a process that creates more orphan blocks can harm the viability of Bitcoin as a method of making transactions.

Invalidating blocks that a large number of people believe to be valid can have serious consequences. The selfish miner may not include the same set of transactions as the rest of the network which will cause problems when the selfish miner releases his longer set of blocks. Transactions that were included by the honest miners, but not by the selfish miner, will be invalidated after the selfish miner releases his blocks. By turning the honest miners' blocks into orphans, the opportunity for a double spend attack is created as a byproduct. In response, merchants may require a larger number of confirmations before accepting a payment which harms the ability to use Bitcoin to make transactions.

Since the consequences of selfish mining can potentially be very large, it is important to detect selfish mining so that the community can be informed and take action against the selfish miner. The authors of the original selfish mining paper suggested that the amount of time taken between consecutive blocks can potentially provide evidence of selfish mining[2]. When the selfish miner publishes their longer chain, the rest of the network would need to hear about those blocks in quick succession in order to switch to the selfish miner's chain. This would result in many cases where multiple blocks appear to be found at the same time.

Previously, Matt Springer attempted to detect selfish mining by using BlockChain.info's API[3]. Whenever Springer received data from BlockChain.info, he recorded that time as the time a block was created. With the data, he was able to record the amount of time it took for the next block to be found. Springer compared the proportion of blocks that were created 2 minutes or less after the previous block to the amount that would be expected based on the exponential distribution. His conclusion was that the observed proportion fell within a 95% confidence interval, so there was no evidence of selfish mining. As far as I know, this is the only data based approach to detecting selfish mining that has been conducted so far.

[1] http://arxiv.org/pdf/1311.0243v5.pdf
[2] http://hackingdistributed.com/2014/01/15/detecting-selfish-mining/
[3] http://scienceblogs.com/builtonfacts/2014/01/11/is-bitcoin-currently-experiencing-a-selfish-miner-attack/

The method that I used to detect selfish mining is similar to Springer's method, but I modified several parts to be more accurate. First, I used BlockCypher as my source of data instead of listening to BlockChain.info's API. The advantage is that there is no additional delay while waiting for BlockChain.info to send data to me after they receive a block. Another advantage is that I can gather data about blocks retroactively instead of waiting to listen to new blocks. By doing so, I was able to quickly gather data about blocks 375,000 through 385,000.

I used the BlockCypher API which contains a block overview method that returns an object that contains a received_time field[4]. The received_time field contains the time that BlockCypher's servers received the block being requested. To find the time taken to find block i, I simply subtracted the time block i-1 was received from the time block i was received. To analyze the data, I chose to split up my data into sets of 1,000 blocks. The reasoning behind the split is that a small set of blocks may not provide enough observations for a statistical test to come to a conclusion. On the other hand, a selfish mining attack may not be attempted for the entire duration of a set. If a set is too large that it captures a period where everyone is mining honestly plus a period that includes selfish mining, the test may fail to find significant results. In the end, the decision of 1,000 block sets is arbitrary.

Since Bitcoin mining is a poisson process, the time between two consecutive blocks will follow an exponential distribution. Therefore, I used a goodness of fit test to measure the deviation between the block times I observed and the expected distribution. In order to do this, I had to divide the block arrival times into buckets. A rule of thumb for the goodness of fit test is to ensure the expected number of occurrences in each bucket is at least 5. My choice was to divide the arrival times into 20 buckets of 100 seconds with one bucket for blocks that took over 2,000 seconds.

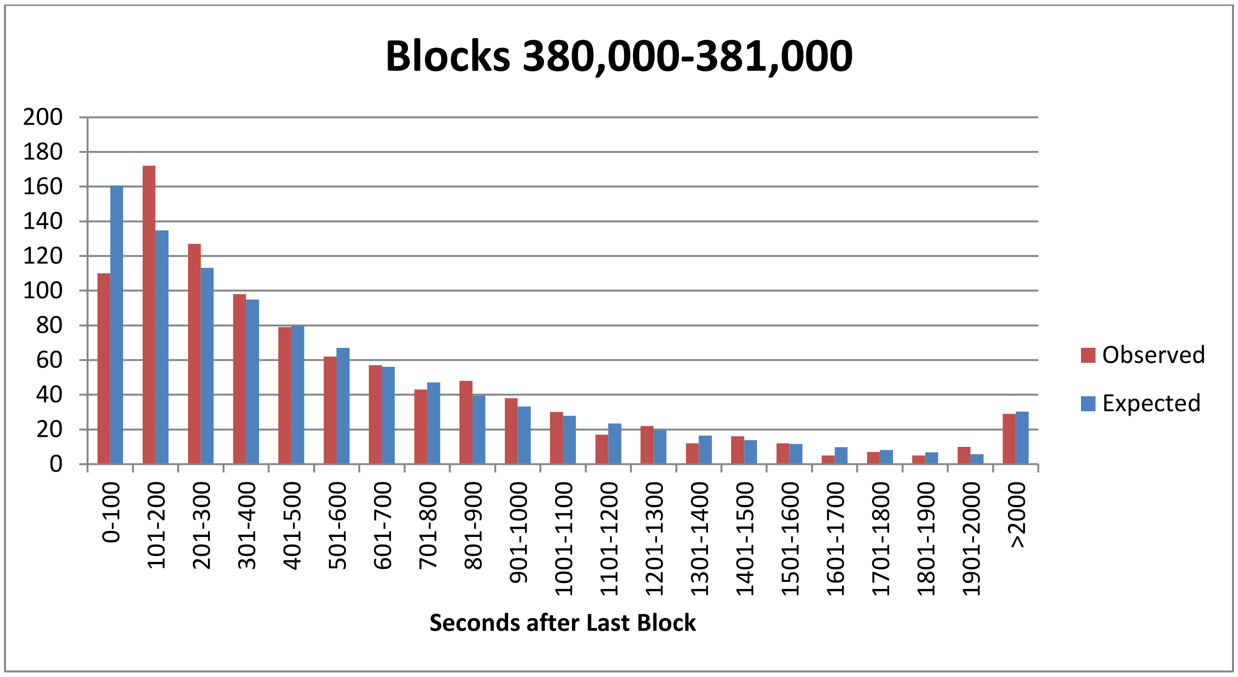
An additional consideration I took into account is the propagation delay of blocks. On average, recent blocks take about 10 seconds to reach 50% of nodes[5], so I decided to use this number as my propagation delay. The key point is that the time BlockCypher receives a block is about 10 seconds after the block has been created. Similarly, miners will have to wait until they receive the next block before they can start working on it. Therefore, I decided to subtract 10 seconds from each observation to adjust for this.

Another factor I took into account was the changing hashrate of the Bitcoin network. The average block time is targeted at 10 minutes, but changes in the hashrate will cause changes in the block times. Instead of using a constant 10 minutes as my average, I used the average of all the block arrival times as my parameter for the exponential distribution.

---

[4] http://dev.blockcypher.com/#block
[5] http://bitcoinstats.com/network/propagation/

The most significant result out of the 10 sets I tested is shown below. The most notable difference is that the number of block times that fall in the 0-100 second bucket is less than what the exponential distribution would predict. It seems that those observations have been shifted into the 101-200 second bucket.



The p value for the test is 0.004. This would indicate that the deviation from the expected counts is not caused by change. However, I conducted 10 tests and this was the one that stood out the most. The chance that any test out of 10 would result in a p value of .004 or less by chance if the distribution was actually exponential is about 4%. There is still a chance that the deviations were caused by chance.

Even if the deviations were not caused by chance, the cause may not necessarily be selfish mining. One of the indicators of selfish mining is an increase in orphaned blocks. However, BlockChain.info reports no orphans being found during that time period[6]. In addition, one would expect that more blocks would appear in the 0-100 second bucket, but the opposite was found. Most likely, there is another reason for this deviation that is not selfish mining. In addition, mining power in the Bitcoin network is fairly decentralized so that the effectiveness of a selfish mining attack would be fairly low. In conclusion, the data shows that some sets of blocks have generation times that do not follow the exponential distribution, but these differences are probably not caused by selfish mining.

[6] https://blockchain.info/orphaned-blocks