CSC2125: Homework #3

Due on October 15

Fan Long

Thomas Hollis

Problem 1

Why withholding a mined block can enable the attacker to gain more rewards? Note that the attacker still expects to generate the same number of blocks.

Solution

Eyal and Sirer explain in their 2014 paper the concept of Selfish Mining. Selfish mining enables an attacking pool to strategically not release mined blocks immediately (i.e. withhold blocks) to gain advantages over other honest pools and users.

Suppose an attacking pool finds the solution to the current PoW hash challenge, i.e. successfully mines the next block in a blockchain. If this attacking pool withholds this mined block, say block A1, then this means that the pool can begin to work on a private branch hidden from the rest of the network. This can be achieved by the attacking pool by mining on top of block A1 (kept private). Since block A1 is private and not released to other pools, this gives the attacking pool an advantage (i.e. a head-start) as they can start working on the PoW of a new block, say A2, (mined on top of block A1) before the other pools.

However, for this attack to be successful we must consider the time when the other honest pools catch up and mine their own alternative solution to block A1. As soon as the others working on the public branch successfully mine the alternative to block A1, say B1, the withheld block A1 is released by the attacking pool immediately. The other honest pools and users will choose either A1 or B1 depending on which block propagates first. This is referred to as the propagation battle, and whichever block wins the propagation battle is the block that will be awarded the mining reward. Indeed, under normal circumstances this would be approximately a 50/50 chance of either the withheld black A1 or the honest block B1. However, to improve their chances the attacking pool unanimously chooses their own withheld block A1. Therefore, in the cases where A1 is chosen over B1 (slightly more likely), then the attacking pool as already started work on block A2. Since the selfish-mining pool is already ahead of the computational race it has slightly more chances of mining the next block A2. Hence, the attacking pool will on average gain more rewards by withholding blocks (selfish mining). This is therefore possible even without increasing the pools share of computational power (i.e. possible while still expecting to generate the same number of blocks).

Problem 2

In the selfish mining strategy, if the attacker always loses the propagation battle, how much computation power does he need for the attack to be profitable? Why?

Solution

The computational power needed for the attack to be profitable in the selfish mining strategy is 1/3 or roughly 33% if the attacker always loses the propagation battle. This is because of the following process:

We know that:

- Attacker percentage of network hash-power: $H_p = X$
- Propagation battle win rate, i.e. portion of network that picks up on attacker chain: $Z = 0$ (as in the question)

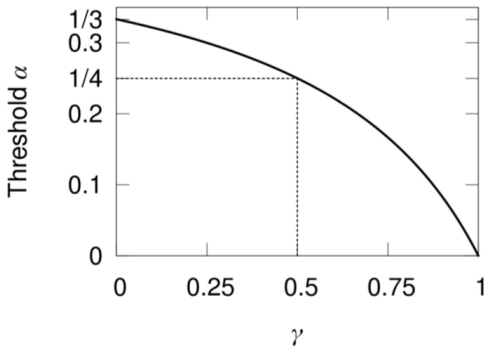
Therefore:

- Block A1: Attacker mining on the private chain (identical to the public chain) and discovers the next block 0A with probability $P = X$. With probability $P = 1 - X$ the network discovers a block (and thus the attacker resets his private chain to match the public chain).

- Block A2: The attacker has mined one block (block A1) and kept it private. With probability $P = X$ he will mine the next block (block A2). With probability $P = 1 - X$ the network will mine the next block (blockB2). If this happens, the attacker releases A1 and hopes he wins the propagation battle.

- Propagation battle: There are now two competing blocks. The attacker will try to mine on top of his own block with a probability $P = X$. The network however will partially work on the attacker's block and partially on the network block. The probability that the attacking block persists (i.e. network finds a block on top of the attacking block) is $P = (1 - X)Z$. Conversely the probability that the network's block persists (i.e. network finds a block on top of their own block) is $P = (1 - X)(1 - Z)$

By inspection, the above information shows that if Z approaches 1 (i.e. if the propagation race is always won) then the attacker will never have to discard a block thus will be mining at his full efficiency while the rest of the network will be mining at partial efficiency. As Z decreases, the attacker's advantage decreases. Eyal and Sirer showed that the attacker becomes more efficient than the public network at $H_p > 1/3$ for any Z. This can be seen in the following figure extracted from their paper:



Indeed, the threshold is therfore $H_p = \frac{1}{3}$.

Problem 3

Consider the fact that most miners now join mining pools to mine cryptocurrencies, whats the economic consequence of the selfish mining vulnerability?

Solution

As most miners now join mining pools, the economic consequence of the selfish mining vulnerability is that mining pools are pressured toward selfish mining. This is because of the free market dynamics of cryptocurrency mining. Indeed, miners are incentivised to join pools that will give them the most reward for their contributed power. This therefore leads to many cryptocurrencies with a high rate of selfish mining and thus leads to a large number of orphaned blocks (note: these blocks are more accurately called extinct or stale blocks).

However, these economic dynamics are more common in alternative cryptocurrencies than in BTC. This is because in BTC there have been many Bitcoin Improvement Proposals (BIPS) which have tried to tackle the selfish mining issue. One such proposal involves randomly assigning miners to various branches when forks occur. Another proposal provides a threshold limit for the reach of mining pools. Finally, another proposal is to discriminate blocks depending on their timestamps so if a long list of blocks are published all at once than the rest of the nodes would weigh their validity lower that they would normally. In addition, there are some counter arguments that suggest that in fact selfish mining is less profitable in BTC than initially expected due to a flaw in the mathematical model assumed (the assumption of independent and identical distribution).

Problem 4

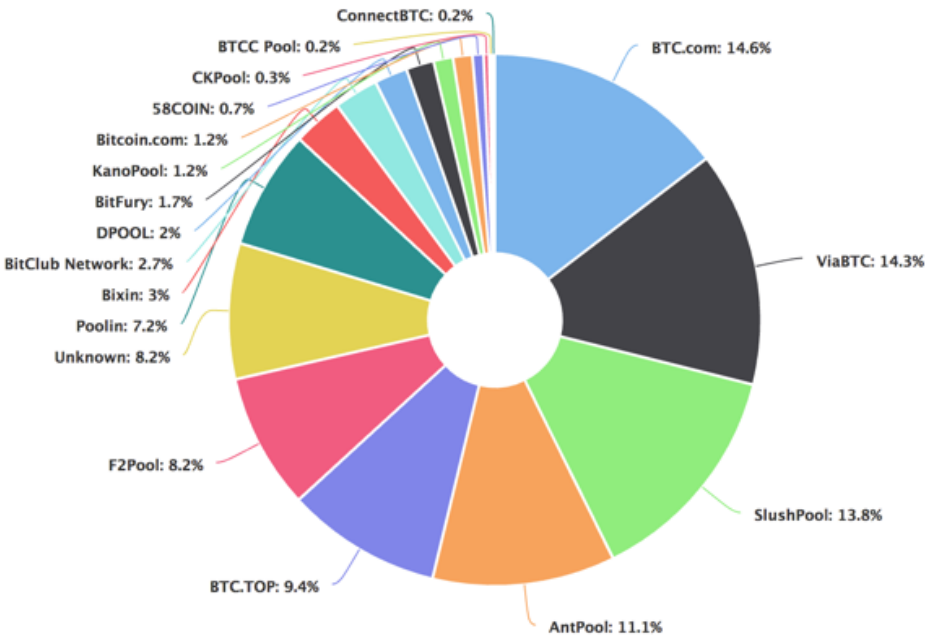
Search the web to survey the mining pool distributions of top cryptocurrencies: BTC, ETH, LTC, XMR, and ZEC. How many of them are vulnerable to selfish mining attacks?

Solution

To determine whether or not a pool is vulnerable to selfish mining we have to make some assumptions. For the sake of this exercise, lets say two of the largest pools can collude to try to reach the 33% critical threshold for selfish mining (for the worst case of Z=0). We also need to assume, as pointed out by Koppelmann et. al. that the value of a 33% selfish mining attack is significantly greater than the value of the mining reward otherwise we would just mine at the 33% level. We also need to assume that the reward from a 33% attack is greater than the mining cost at that moment. It is worth mentioning that only mineable cryptocurrencies are theoretically vulnerable to selfish mining attacks and all the investigated cryptocurrencies are mineable thus have the potential to be vulnerable.

Now we can look at the hash power distribution by mining pool of all these potentially vulnerable cryptocurrencies to assess their resistance to selfish mining attacks.

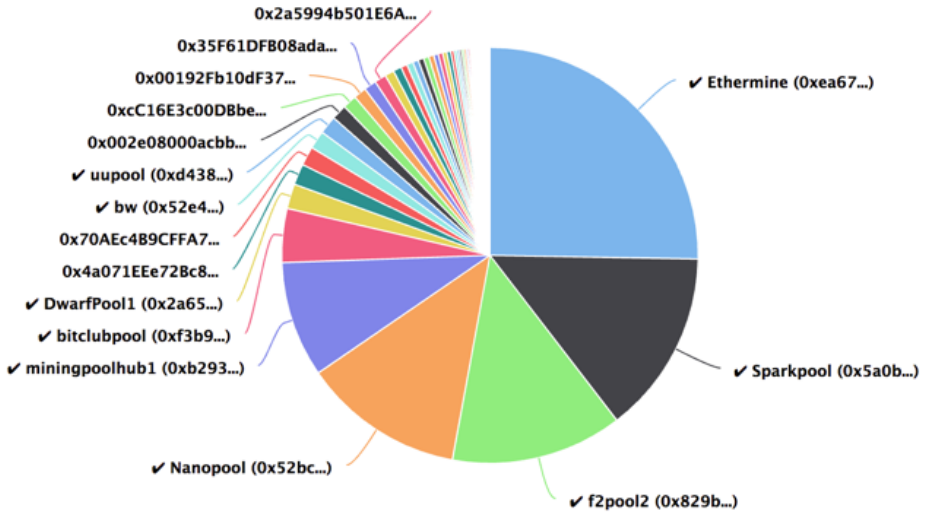
The main BTC mining pools at the moment (last 4 days) are BTC.com (14.6%), ViaBTC (14.3%), SlushPool (13.8%), AntPool (11.1%), BTC.TOP (9.4%), F2Pool (8.2%), Unknown (8.2%), Poolin (7.2%), Bixin (3%), BitClub Network (2.7%), DPOOL (2%), BitFury (1.7%), KanoPool (1.2%), Bitcoin.com (1.2%), 58COIN (0.7%), CKPool (0.3%), BTCC Pool (0.2%) & ConnectBTC (0.2%). Note these percentages change over time but the BTC mining pool space is still quite segmented.



`https://www.blockchain.com/en/pools`

Therefore, since the top two pools only sum to a combined network power of 28.9%, I would argue that BTC is currently relatively resistant to selfish mining attacks from pools. This argumentation is also backed up by previous information written in this document regarding the existence of BIPS attempting to protect BTC against selfish mining attacks.

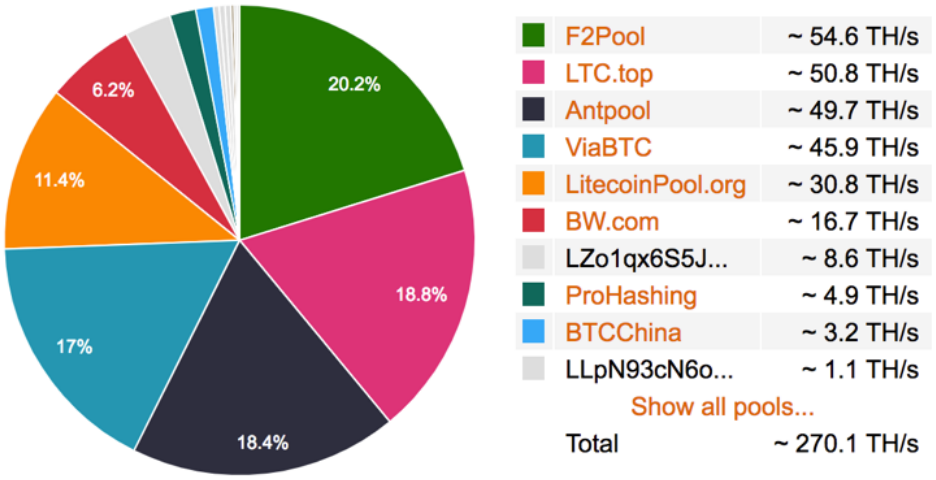
The main ETH mining pools at the moment (last 24hrs) are Ethermine (25.3%), Sparkpool (14.3%), f2pool2 (13.3%), Nanopool (12.6%), miningpoolhub1 (9%), bitclubpool(4.2%), DwarfPool1 (1.9%), many other smaller pools (19.4%).



`https://www.etherchain.org/charts/topMiners`

Since the two largest ETH pools combined can easily reach the 33% threshold, I would argue that ETH is moderately vulnerable to selfish mining attacks (this is because the assumptions made here are not necessarily the most robust).

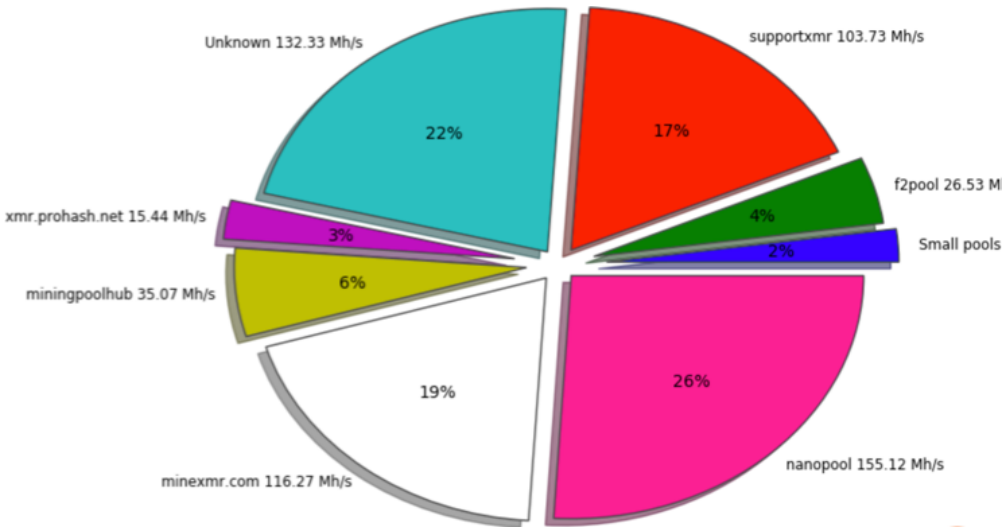
The main LTC mining pools at the moment (last 20 hours) are F2Pool (20.2%), LTC.top (18.8%), Antpool (18.4%), ViaBTC (17%), LitecoinPool.org (11.4%), BW.com (6.2%), rest (8%).



`https://www.litecoinpool.org/pools`

Since the two largest LTC pools combined can easily reach the 33% threshold, I would argue that LTC is moderately vulnerable to selfish mining attacks.

The main XMR mining pools at the moment (on 22 Sept 2018) are nanopool (26%), Unknown (22%), Minexmr.com (19%), supportxmr (17%), miningpoolhub (6%), f2pool (4%), xmr.prohash.net (3%). rest (2%).



`https://www.minexmr.com/pools.html`

Since the two largest XMR pools combined can easily reach the 33% threshold, I would argue that XMR is moderately vulnerable to selfish mining attacks.

The main ZEC mining pools at the moment (last 6 hours) are Flypool (20%), F2Pool (19%), Poolin (18%), many small others (43%)

`https://explorer.zcha.in/statistics/miners`

Since the two largest ZEC pools combined can easily reach the 33% threshold, I would argue that ZEC is moderately vulnerable to selfish mining attacks.

Problem 5

Describe the rationale of the block withholding attack. Why this can be a strategy for a large mining pool to attack small mining pools.

Solution

The block withholding attack (a.k.a. Lie In Wait attack) is based on the concept of withholding a block within a pool. Essentially, Lie In Wait is achieved when a miner delays submitting blocks that he found to the pool that he is part of. He uses the knowledge of the imminent block to focus his mining where it is most rewarding.

In practice, this is implemented in a fairly straightforward way. The attacker starts off in a "wait phase" in multiple pools where he mines at the same hashrate in each. However, once the attacker finds a block he directs all his mining capacity to the pool where it is found and only submits the block after a given time T. During this time, the attacker attempts to find as many shares (partial solutions) as possible generating extra gains. After this period of time T, the attacker submits the whitheld block and goes back to waiting across all the pools.

This can be a strategy for a large mining pool to attack small mining pools because a large mining pool can build a proxy server to tunnel part of its hash rate into smaller pools, joining as a peer of these smaller pools. Then from this position the attacking pool can carry out block witholding attacks across all the pools it is currently waiting in as soon as this attacking pool finds the next block.

Problem 6

In reality, selfish mining does not happen very often but the block withholding attack happens a lot. Whats the reasons behind this?

Solution

Multiple people have criticised Eyal and Sirers (2014) initial selfish mining claim, including C.S. Wright in his paper The Fallacy of the Selfish Miner in Bitcoin: An Economic Critique (2018). Indeed, Wright explains that the Selfish Miner strategy poses no threat to the BTC protocol. The reason behind this is that for any given difficulty, the SM strategy results in fewer blocks claimed for the same investment cost, thus profit will be lower than when using the default strategy. It is for this reason that we can see that selfish mining in fact does not happen very often as it is simply not particularly economically profitable.

On the other hand, block-withholding within a pool (a.k.a. Lie In Wait attacks) are more common. This is because they are easy to pull off (a simple change to mining strategy) and are demonstrably more profitable than regular pool mining. In addition, many protocols have not yet protected themselves against this malicious behaviour as the suggested solution called "oblibvious shares" remain unimplemented for most blockchains. Finally this attack is very difficult to detect as it just appears to be ordinary bad luck.

Problem 7

Any other questions you want to discuss in the class?

Solution

- Are there any notable examples of Finney attacks in Bitcoin? If not, why?
- Are there other examples of Finney attacks in other cryptocurrencies?
- Could we develop an algorithm for examining the rate of orphaned (extinct/stale) blocks to help secure cryptocurrencies? If so has this already been done in an alternative crypto