A Lower Bound on Miner Rewards

Kevin Lu*

April 23, 2019

Abstract

We examine the security of proof-of-work networks against double spend attacks. The original Bitcoin whitepaper examined this from a technical perspective assuming that the attacker is unable to acquire more hash rate than the rest of the network. Here we examine the problem from an economic perspective. Assuming that one *can* attack the network, under what conditions is it profitable to effectuate such an attack? Using this as our criterion for security, we derive a lower bound on the amount that miners must be rewarded to prevent double spend attacks. This work was motivated by a recent working paper by BIS [1].

Disclaimer and Disclosure

The author prepared this report in his personal capacity, not as an employee or on behalf of BKCM LLC. The views contained in the paper are the author's own and do not necessarily reflect the views of BKCM LLC. Nothing contained in the paper should be construed as investment advice. The analysis contained in the paper is theoretical and for informational purposes. The author does not condone performing any double-spend attack referenced in the paper. BKCM holds positions in digital assets, which may be subject to double-spend or other types of attacks.

*Head of Systematic Trading, BKCM LLC, kevin@bkcm.co

1 Introduction

BIS recently published a report [1] that studied the 'economic finality' of proof-of-work blockchains.[1] Essentially, economic finality considers a transaction secure if it is unprofitable to attack it. While we do not agree with several conclusions in their report, this idea captured our interest. We reapply the idea of economic finality with certain different assumptions and interpretations.[2]

The original Bitcoin whitepaper [2] evaluates the probability of an attacker successfully double-spending assuming we know the attacker's hash power[3]. This analysis is used to compute how many blocks we should wait before considering a transaction final (i.e. the probability of the transaction being undone is below some reasonable threshold). For example, suppose we take a less than 0.1% probability of a transaction being undone as our threshold for finality. Then, assuming the attacker has 10% of network, we should wait 5 blocks after a transaction to consider it final. If, instead, we assume that the attacker has 20% of network hash power, we must wait 11 blocks for the same 0.1% confidence. Notably, if the attacker has more than 50% of hash power, then we can never be sure that a transaction is final.

This analysis leaves open the question of how much hash power an attacker might have or acquire. Economic finality gives us a natural method to answer the question. Attackers will acquire hash power as long as the attack is profitable.

This is purely a theoretical analysis for informational purposes. We do not condone actually performing any of the following.

To that end, consider this specific realization of the general double spend attack analyzed in [2]:

1. The attacker borrows H coins that settle on chain at block $b-1$;
2. The attacker then sends H coins to an exchange in block b and begins mining a double spend chain in secret (which does not include this deposit);
3. At block $b+W$, the deposit is cleared to trade. The attacker liquidates the position to cash and withdraws the cash; and
4. The attacker covers the original borrow H in its double spend chain in block $b+W+1$ and releases the double spend chain to the public[4]

Under what conditions is this profitable? The revenue is the double spend at step 3 plus the miner rewards from the double spend chain received in step 4.

[1] We assume some familiarity with proof-of-work blockchains. Please see [1] for an excellent introduction.

[2] And so some sections of this report mirror sections in [1] with changes in notation.

[3] More precisely, the paper assumes we know the ratio of probabilities of an attacker vs an honest node finding the next block, which should be proportional to the amount of hash power the attacker has relative to the rest of the network.

[4] We do not include borrow fees and transaction fees related to moving the borrowed coins in this analysis for simplicity.

The cost is the cost of mining the double spend chain. Below, we examine these more closely.

To keep things simple, we consider only the case where the attacker tries to obtain more than 50% of hash power. Also, for simplicity, we consider only this specific attack.[5][6]

2 Cost of Attack

We assume the exchange waits W blocks before allowing anyone to trade newly deposited coins. So the cost of attack is the cost of mining $W+1$ blocks before the main chain does so. In order for the attacker to be reasonably confident[7] that it will mine $W+1$ blocks *before* the main chain, it must rent (significantly) more hash power than the rest of the network combined.

Let the per hash cost of this exorbitant amount of hash power[8] be denoted by C_b^r. Then the cost of the attack is C_b^r times the total expected number of hashes required to mine $W+1$ blocks. Let D_i be the expected number of hashes require to mine block i, i.e. its difficulty, then

$$\text{Cost of Attack} = C_b^r \sum_{i=b}^{b+W+1} D_i \tag{1}$$

The difficulty, D_i, is opaque; each blockchain has its own formula and schedule for updating it. In the next section we derive a more transparent formula for difficulty.

2.1 Equilibrium Difficulty

We turn to an economic argument to create a universal formula for difficulty.

[5] True economic finality would require *all* attacks to be unprofitable. Since we are examining only one specific attack, we are only deriving one possible bound. We leave studying other attacks to future work. For example, one interesting attack is a 'long-only' version where the attacker must buy coins instead of borrowing them.

[6] Finally, for simplicity from a mathematical rigor perspective, we do 'hand-wave' over some technical details in a few places (mostly by using expected values in place of random variables, which is equivalent to assuming that all relevant actors are risk neutral).

[7] The attacker must deal with the same double spend probabilities studied in [2], described in the introduction. For example, for the attacker to mine five blocks before the main chain with 99.9% probability, the attacker must have 90% of the total hash rate of the network. Thus, the attacker would need to rent **nine times** more hash power than the rest of the network combined. This presents an interesting optimization problem for the attacker; the attacker must balance cost of renting hash power with length of the double spend side chain. A more complete analysis would need to consider this. Here, we just assume there is some threshold probability that the attacker is comfortable with.

[8] Cost is usually a convex function of quantity for any widget. So every extra hash that the attacker rents is going to cost more than the last, and in particular they will all cost more than the current average cost C_b. In short, we want to emphasize that we expect C_b^r to be huge relative to C_b, but this is sort of hidden in our notation

The revenue of mining block b is the current block reward R_b plus fees for all of the transactions in that block $\sum_{t \in b} f_t = F_b$, all multiplied by the price of the token, P.

$$Revenue_b = P(R_b + \sum_{t \in b} f_t) = P(R_b + F_b)$$

The expected cost of mining that block is the cost per hash at the time, C_b, times the difficulty of the block, D_b, (similar to above)

$$Cost_b = C_b D_b$$

Now we make our key assumption: $Revenue_b = Cost_b$. This should be true if mining is competitive and the mining market is at an equilibrium.[9]

$$\begin{aligned} Revenue_b &= Cost_b \\ P(R_b + F_b) &= C_b D_b \\ D_b &= \frac{P(R_b + F_b)}{C_b} \end{aligned} \tag{2}$$

Combining equation 1 and equation 2, we get

$$\begin{aligned} \text{Cost of Attack} &= C_b^r \sum_{i=b}^{b+W+1} D_i \\ &= \frac{C_b^r}{C_b} P \sum_{i=b}^{b+W+1} (R_i + F_i) \end{aligned}$$

3 Gain of Attack

If the attack is successful, the attacker gains two things:

1. Proceeds from liquidating the double spend coins; and
2. Mining rewards from the double spend chain

However, once the attack is revealed, we expect some damage to trust in the network, which may depress market prices. Furthermore, the network may initiate a hard-fork specifically to undo the double spend attack.

We capture these two effects with two more variables.

[9] One sticking point is that the attack will require renting **a lot** of hash power (see footnote 7). This would push the mining market out of an equilibrium; it is not clear if equation 2 will still hold under the attack scenario. Since the double spend chain is mined in secret, the attacker's rented hash power will not be visible on-chain until after the attack is complete. If the attacker can rent the hash power without tipping off other miners, then perhaps the rented hash can be achieved without disturbing the equilibrium. However, the attacker still faces the difficulty adjustment algorithm on its double spend chain. Some chains update difficulty relatively infrequently so this is not an issue. For example, Bitcoin only adjusts difficulty every two weeks, whereas the double spend attack would probably complete in a few hours. Chains with frequent difficulty updates would require some optimization, see footnote 7.

1. Let π be the probability that the network hard-forks to undo the double spend. We assume that the exchange is somehow able to clawback the proceeds from the liquidation under this scenario.[10]

2. Let P_A be the market price after we reveal our attack. Presumably P_A is signficantly less than P.

Consider first the case of no hard-fork. Note that the double spend coins are liquidated *before* the attack is revealed. The attacker has H coins to double spend. Assume the attacker receives a volume-weighted average price of P for them, which we expect to be close to the prevailing market price at the time. The attacker also receives all of the mining rewards from its double spend chain but at a market price of P_A.

$$\text{Gain of Attack if no hard-fork} = PH + P_A \sum_{i=b}^{b+W+1} (R_i + F_i)$$

If the network does hard-fork, the attacker returns its borrowed coin and gains nothing (recall we assume no borrow fees).

$$\text{Gain of Attack if hard-fork} = 0$$

Combining the two with probability π:

$$\text{Expected Gain of Attack} = (1-\pi)\left(PH + P_A \sum_{i=b}^{b+W+1} (R_i + F_i)\right)$$

4 Profitability of Attack

Finally, we can evaluate the profitability of this double spend attack. In order for transactions on the chain to have economic finality, this attack must be unprofitable:

$$\begin{aligned}
&\text{Profit of Attack} = \text{Expected Gain of Attack} - \text{Cost of Attack} \\
&\text{Profit of Attack} < 0 \\
&\text{Cost of Attack} > \text{Expected Gain of Attack} \\
&\frac{C_b^r}{C_b} P \sum_{i=b}^{b+W+1} (R_i + F_i) > (1-\pi)\left(PH + P_A \sum_{i=b}^{b+W+1} (R_i + F_i)\right)
\end{aligned}$$

[10] Whether this is through the traditional legal system or in some new smart contract system (and therefore automatic), it does not matter for the purposes of this paper. The results of this paper can be easily extended to the case where the proceeds from the liquidation cannot be clawed-back.

Isolating the total mining revenue:

$$\frac{C_b^r}{C_b} P \sum_{i=b}^{b+W+1} (R_i + F_i) > (1-\pi)\left(PH + P_A \sum_{i=b}^{b+W+1} (R_i + F_i)\right)$$

$$\frac{C_b^r}{C_b} P \sum_{i=b}^{b+W+1} (R_i + F_i) > PH + P_A \sum_{i=b}^{b+W+1} (R_i + F_i) - \pi PH - \pi P_A \sum_{i=b}^{b+W+1} (R_i + F_i)$$

$$\frac{C_b^r}{C_b} P \sum_{i=b}^{b+W+1} (R_i + F_i) - P_A \sum_{i=b}^{b+W+1} (R_i + F_i) + \pi P_A \sum_{i=b}^{b+W+1} (R_i + F_i) > PH - \pi PH$$

$$\left(\frac{C_b^r}{C_b} P - P_A + \pi P_A\right) \sum_{i=b}^{b+W+1} (R_i + F_i) > PH - \pi PH$$

$$\left(\frac{C_b^r}{C_b} - \frac{P_A}{P} + \pi \frac{P_A}{P}\right) \sum_{i=b}^{b+W+1} (R_i + F_i) > H - \pi H$$

$$\sum_{i=b}^{b+W+1} (R_i + F_i) > H(1-\pi)\left(\frac{C_b^r}{C_b} - (1-\pi)\frac{P_A}{P}\right)^{-1}$$

Currently, we are measuring the attacker's borrowed coins H in number of coins. Let us change units so we measure the borrowed coins as a fraction of the total supply of coins instead. So the attacker borrows some fraction, denoted by p, of the total supply at block b, denoted by T_b, $pT_b = H$.

Evidently, we must have $0 \leq p \leq 1$, but also (and more interestingly) p is perhaps related to how decentralized the network is. Presumably it is harder to borrow a large fraction of the coins if they are held by many different parties. So decentralization places an upper bound on p.

$$\sum_{i=b}^{b+W+1} (R_i + F_i) > pT_b(1-\pi)\left(\frac{C_b^r}{C_b} - (1-\pi)\frac{P_A}{P}\right)^{-1}$$

$$\frac{1}{T_b} \sum_{i=b}^{b+W+1} (R_i + F_i) > p(1-\pi)\left(\frac{C_b^r}{C_b} - (1-\pi)\frac{P_A}{P}\right)^{-1}$$

Finally, to make one last simplification, let us assume that the block reward does not change and the fees are evenly distributed between blocks during the attack period, so $R_i = R_b$ and $F_i = F_b$.

$$\frac{1}{T_b} \sum_{i=b}^{b+W+1} (R_b + F_b) > p(1-\pi)\left(\frac{C_b^r}{C_b} - (1-\pi)\frac{P_A}{P}\right)^{-1}$$

$$\frac{1}{T_b}(W+1)\,(R_b + F_b) > p(1-\pi)\left(\frac{C_b^r}{C_b} - (1-\pi)\frac{P_A}{P}\right)^{-1}$$

$$\frac{R_b + F_b}{T_b} > (W+1)^{-1} p(1-\pi)\left(\frac{C_b^r}{C_b} - (1-\pi)\frac{P_A}{P}\right)^{-1} \quad (3)$$

Equation 3 is our main result. The left hand side is the current per block miner reward as a fraction of the current total supply. This equation gives us a lower bound on how much we need to reward miners in order to prevent double spending attacks.

The pieces of the right hand side of equation 3 seem intuitive:

1. Bigger W makes the right hand side smaller, so we can pay miners less by waiting longer
2. Larger π makes the right hand side smaller, so the more willing the network is to undo double spends, the less attractive double spends are
3. A larger $\frac{C_b^r}{C_b}$ makes the right hand side smaller, so more expensive rental hash makes double spending less attractive
4. A smaller $\frac{P_A}{P}$ makes the right hand side smaller, so if a double spend attack has a large price impact, then double spending is less attractive

Some details of the right hand side are very interesting:

1. Even if $P_A = 0$, i.e. a double spend attack destroys the network, the right hand side is still positive; the double spend could still be profitable;[11]
2. The only way to make the right hand side zero is to have $p = 0$, i.e. impossible to borrow, or $\pi = 1$, i.e. always hard-fork;
3. The right hand side tends slowly to zero in terms of W. This is in contrast to the Nakamoto result where the probability of double spend tends quickly (exponentially) to zero in terms of W; and
4. None of the terms have an outsized effect on the final result relative to the others (e.g. nothing is squared or in an exponent). So there is not an obvious thing to optimize (although W is the main thing under our control).

5 Try some numbers

Unfortunately, many of the quantities here are not directly observable. For example, to execute such an attack on Bitcoin we would need to acquire many ASICs which would likely distort their prices in unpredictable ways; thus C_b^r is difficult to estimate. However, Ethereum Classic (ETC) is a special case where we can reasonably estimate all of the quantities which we detail below. The motivated reader can easily create a spreadsheet to test various assumptions on other blockchains.

The current block reward is 4, fees are near zero, and current supply stands at about 108 million, giving us a left hand side of 3.7×10^{-8}. There was an

[11] Perhaps this actually is not so interesting. This would be equivalent to short-selling the token and then destroying the network

apparent double spend attack around Jan 7, 2019. A few days later market price fell by about 20% so let $\frac{P_A}{P} = 80\%$. Given the events surrounding the creation of Ethereum Classic, we think it is safe to assume that $\pi = 0$. Kraken currently has a wait time of $W = 120$ for ETC. Hash power for Ethereum Classic seems relatively easy to get at Nicehash, let's estimate $\frac{C_b^r}{C_b} \approx 3$.

The main difficulty is estimating the size of the borrow market for ETC. Let's guess we can easily borrow around 250 thousand ETC which gives $p = 0.0023$. With these we get a right hand side of 8.6×10^{-6} which is less than our previously computed left hand side, so ETC is still vulnerable to a double spend. A lot of work for an apparently obvious result.

But we can start to see the magnitude of the problem. If we backout W to make the right hand side equal to the left hand side, leaving everything else the same, we need $W \approx 28000$ (assuming a block time of 15 seconds, that is 4 days!). If we instead backout $\frac{C_b^r}{C_b}$, we find that we need $\frac{C_b^r}{C_b}$ to be close to 600, ie we need it to be almost 200 times harder to rent ETC hash power! Finally, if we instead backout R, the block reward necessary, we get $R \approx 1000$, an annualized inflation rate close to 2000%!

6 Conclusion

We have derived a lower bound on the amount of rewards miners must receive to disincentivize a particular double spending attack under many assumptions. On an immediate practical basis, this bound is not very useful; see any of the many footnotes for areas for improvement and future research. However, we hope further research along these lines can yield a strong theoretical baseline for blockchain incentives and thus lead to more secure blockchains in the future.

References

[1] Raphael Auer. *Beyond the doomsday economics of "proof-of-work" in cryptocurrencies.* BIS Working Papers, No 765, 21 January 2019. `https://www.bis.org/publ/work765.htm`

[2] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* `https://bitcoin.org/bitcoin.pdf`