6 Censorship attacks

Censorship attacks can happen in a number of ways. We are going to explore what portion of the hashrate is necessary to perform an attack and how that influences the fee that must be paid in order to overcome the incentives against including the transaction on the blockchain.

The least effective way for a pool or collection of pools to censor is to simply to decline to process a transaction. Even if the pool has a majority of the hashrate, anything less than 100 % hashrate will allow the transaction on the chain. This is why Bitcoin is often described as *censorship-resistant*. Effective censorship requires a more aggressive strategy. To effectively censor, the censoring parties should commit to supplanting any blocks that process forbidden transactions. If the censoring parties have more than 51 % they should be able to accomplish this in the long run, with probabilities as described in Chapter 4. If the censoring pool has less than 51 % of the hashrate, they should not expect to be able to overwhelm the rest of the hashrate on a consistent basis, at least by brute force. Even if they accomplish this feat once, or twice, the transaction will stay in the mempool and should eventually be confirmed.

The most effective approach for the censoring party is to announce their intention to attempt to supplant any blocks containing undesirable transactions and then allow incentives to influence other rational miners. At this point, presuming the censors can be taken seriously with this commitment, this promise will be taken into consideration by other miners. Before getting into the details, we offer a quick example. Suppose that a pool with 25 % of the hashrate announces that they will attempt to uncle any blocks which contain transactions which spend a certain output. Assume this threat is received as credible by other miners on the network. Now suppose a transaction appears in the mempool. The non-censoring miners have two options: leave it alone or attempt to mine it. If the transaction fee is small or non-existent, the miners have little or nothing to gain by including the transaction when attempting to construct a block. On the other hand, if they do include the transaction in their next block, there will be a risk: There is at least a 1/16 chance that the censoring pool will immediately write two blocks on top of the latest compliant block, before anyone appends to the offending block. So the expected value for including such a transaction in a block is at least 1/16 of a block reward less than leaving the transaction alone. In order to compensate for this the transaction should include a fee of at least 1/16 of a block reward or expect that miners are either ignorant of or resistant to the censorial pool's threats to supplant the block.

In practice, an entity with a bit of cash on hand who wanted to censor transactions may try the approach of committing to rewarding miners who supplant offending blocks. It is easy to choose the reward large enough so that rational miners will be induced to pursue a reorg whenever an offending transaction is included in a block. By backwards induction, rational miners will simply choose not to include that transaction, unless the fees are large enough to merit the risk. Of course the censor is likely to be a party who can simply bid higher, to the point that the fee required to spend

https://doi.org/10.1515/9783110772845-006

the transaction may be more than the unspent output. Another practical way to censor transactions is to have major financial institutions agree that the outputs from certain UTXOs are *tainted*. If such institutions refuse to treat these UTXOs and their progeny as valid Bitcoin, the effect could cascade throughout the ecosystem: Parties are unlikely to accept an output if they do not believe they will be able to trade it for something of value.

We model a case where a subset of miners tries to practice censorship by making a commitment to mining only on certain chaintips. This can happen in practice if a group of mining corporations resides in a jurisdiction that can exercise control over the miners, in which case the authority can declare it illegal to participate in the settlement of certain transactions. In this case "settlement" could mean mining a chaintip in which such transactions are less than three blocks deep. We assume that the miners will continue to mine the old chain, instead of turning off machines (it is possible that a miner who wishes to remain law-abiding without actively participating in censorship could just refuse to participate in reorgs, satisfying authorities by idling machines until the transactions are considered settled).

We consider three sets of mining pools. The censorial (C) mining pool will have hashrate p and is not interested in minimizing costs. They will follow a fixed predetermined strategy, with the primary objective of keeping offending transactions off of the blockchain. There may also be a non-compliant (NC) pool, which has hashrate q, which is not interested in censorship for ideological or technical reasons and hence will always follow the longest-chain rule and will mine any transactions. Finally there will be a rational (R) set of pools. These may have different sizes $\varepsilon_1 + \cdots + \varepsilon_n = r$ and are considered economic free agents: They will join the compliant pool when more profitable and proceed with non-compliant mining when profitable.

The NC pool will be assumed to not use any strategy: simply mine the longest chain. The C pool will declare their strategy at the beginning, which will be simple, and other players in the game will take this to be credible. For example, the C pool can declare that they will attempt to supplant all non-compliant blocks that are no more than k blocks deep, but will not pursue reorgs beyond a certain depth. It is then left to the remaining rational pools to determine their respective strategies.

If the C pool owns 51 %, they control the network. They can uncle any block, eventually, as is their prerogative. If the C pool has less than 50 % it makes sense to use a loss-cutting strategy. There is no sense pursuing an increasingly unlikely attempt to censor a specific transaction indefinitely.

6.1 Working example: two rational pools

Suppose we have only two rational pools of sizes $\varepsilon_1, \varepsilon_2$ so that

$$p + q + \varepsilon_1 + \varepsilon_2 = 1.$$

We assume that p is less than 50 %. Suppose that C declares that their strategy is to attempt to uncle any offending blocks that are less than k blocks deep. If k blocks behind, they will drop the fight and look for a more recent forking point or mine the chaintip with compliant transactions. With this declared strategy we will attempt to begin to build a payoff table for a few possible basic strategies for the two rational pools. For simplicity, assume that the distasteful transactions are happening somewhat infrequently, so that we do not expect to see a series of consecutive blocks containing offending transactions that may demand a more sophisticated censoring strategy.

The game starts when the discountenanced output holder attempts to send a transaction. The unspent output (fee) will be given by λ in units of the block reward. We will assume that NC has positive hashrate q. This assumption means that eventually the NC will win k consecutive blocks and the transaction will be accepted. In practice, depending on the strategies of the rational pools and the size of NC, this terminal event may occur within an hour, or it may take centuries. However, it is a convenient mathematical assumption as we can then model the game as a game that terminates in finite time with probability 1.

Remark 6.1. As a side note, we point out that there is a huge mathematical difference between the words "finite" and "bounded." Unfortunately these words are often used interchangeably by non-mathematicians. For example, saying the supply of a currency is always finite is usually a true statement, whereas saying the supply is "bounded" (as specified in Bitcoin's whitepaper) is a completely different statement. "Bounded" requires a number that provides the upper bound. Similarly, saying that a game terminates in finite time with probability 1 can be true, even if the expectation of the time it takes is infinite. The payoff in any play of the St. Petersburg paradox game (Exercise 2.3) is finite, but not bounded.

From a rational pool's perspective, there are two questions to be asked:

1. What is the expected number of block rewards that a rational pool would waste in fighting the censors?
2. What is the probability that they receive the extra reward from the fee?

Multiplying the answer to the latter by λ and subtracting the former serves as a useful metric in comparing different strategies.

We can think of the game as occurring in a series of rounds. Each round begins when the offending transaction is mined and terminates in one of two ways. The round may terminate with the compliant chain pulling a full block ahead, in which case the game continues. The NC pool attempts to mine the offending transaction on the chaintip, instead of making a come-from-behind attempt. The next round of the game starts when the transaction is again mined on the chaintip. The round may also terminate with the NC chain pulling ahead k blocks. In this case, the round terminates, and so does the game. So to compute the expected outcome for a given pool, we divide the analysis into these two cases.

In each of the scenarios determined by the pools' strategies, there will be a function $w(x)$, which must be computed, that determines the probability that the censorial pool will win the round. When the transaction is mined and the round begins, the probability that the game will continue beyond that round is $w(1)$. The probability that the game terminates is $1 - w(1)$. We can compute the expected value that a pool will have won in the latter case. In the former case we will need to set up some recursion relationships to complete the computation.

In particular, after fixing a strategy regime and a Pool i, define a random variable $W_i \in \{0, 1\}$ via

$$W_i := 1 \text{ if Pool } i \text{ claims the fee } \lambda \text{ when the game ends,}$$
$$W_i := 0 \text{ otherwise}$$

and a random variable $L_i \in \mathbb{Z}_+$,

$$L_i := \text{blocks mined by Pool } i \text{ that have become stale when the game ends.}$$

Then we define the random variable (dropping subscripts when clear)

$$f := \lambda W - L.$$

The goal then is to compute

$$\mathbb{E}[f] = \lambda \mathbb{E}[W] - \mathbb{E}[L].$$

Now

$$\mathbb{E}[f] = \left\{ \begin{array}{c} (1 - w(1))\mathbb{E}[f \mid \text{C loses round 1}] \\ +w(1)\mathbb{E}[f \mid \text{C wins round 1}] \end{array} \right\}, \tag{6.1}$$

and then

$$\begin{aligned} \mathbb{E}[f \mid \text{C wins round 1}] = {} & \big(1 - w(1)\big)\mathbb{E}[f \mid \text{C loses round 2}] \\ & + w(1)\mathbb{E}[f \mid \text{C wins round 2}]. \end{aligned} \tag{6.2}$$

Now define the following quantity:

$$\tau := \mathbb{E}[\text{blocks lost by Pool } i \text{ in any given round} \mid \text{C wins that same round}],$$

which is a constant that does not depend on the particular round. Noting that looking forward, the game looks the same at the start of every round, we can write

$$\mathbb{E}[f \mid \text{C wins round 1}] = \mathbb{E}[f] - \tau. \tag{6.3}$$

We also use notation

$$\sigma = \mathbb{E}[f \mid \text{C loses round 1}]$$

and combine (6.1) with (6.3) to get

$$\mathbb{E}[f] = \big(1 - w(1)\big)\sigma + w(1)\big(\mathbb{E}[f] - \tau\big),$$

which reduces to

$$\mathbb{E}[f] = \sigma - \tau\frac{w(1)}{1 - w(1)}. \tag{6.4}$$

In the following sections, our goal will be to compute the values σ, τ and the function $w(x)$.

6.1.1 Consideration for this choice of function

One difficulty in modeling a game like this is coming up with an appropriate quantity to measure. We have made the decision to essentially model the game as a one-shot affair. The miners in question will mine until the game is over and then return to mine as they had been before the game started. In reality, this may or may not be optimal. In particular, we are not considering situations in which new offending transactions will pop up while a previous transaction is being fought over.

The goal of the computations will be to demonstrate that, given a fee and a hashing power of the censorial pool, the rational pools can compare different strategies. The results will be meaningful with the assumption that the game will be ending in a reasonably short period of time (this is not an unreasonable assumption).

For this reason, the values that appear in forthcoming payoff tables involve only the expected gain or loss from what would be mined in that time period, and do not consider how much time is taken to complete the game. When discussing incentives for mining games, especially those that may be repeated, the *revenue ratio* (expected revenue per expected time, cf. [21, §3]) can be more instructive, but the computation can become more intricate.

6.1.2 There are many strategies

We focus on strategies that are easier to describe and implement.

1. (Non-compliant) First, there is the non-compliant-at-all-times strategy. The pool simply ignores the censors and includes the transaction, and always mines the longest chain using the first-seen rule.

2. (Mildly compliant) A second strategy would be to attempt to build on the censoring fork when the difference is zero blocks but not when it is one block deep. In particular, the pool would never mine the offending transaction, always leaving it be while in the mempool, and will break ties for the censors.
3. (More compliant) A third strategy would be to not only exclude the transaction, but also attempt to build on the censoring fork when the difference is zero or one block and go back to the longest chain if two blocks deep.
4. (Neutral/selective) The pool may decide to abstain from mining when the difference is one block deep, not wanting to risk the event this block would become an uncle. If the energy cost is significant, these sorts of strategies may come into play.

There are many more variations. Computing outcomes with each of these would give us a 4×4 payoff table with 16 outcomes. Even with symmetries, this would already lead to a somewhat lengthy computation, so we restrict ourselves to the first two strategies. This will result in a 2×2 payoff table with essentially four computations, due to symmetries. Our attainable goal is to populate the following table:

Pool 1 strategy \ Pool 2 strategy	Non-compliant	Mildly compliant
Non-compliant	?	?
Mildly compliant	?	?

6.1.3 Case 1: Pool 1 and Pool 2 are both non-compliant

We consider the game from the perspective of Pool 1; the payoff to Pool 2 computation would be identical. From Pool 1's perspective, there is no difference between Pool 2 and NC, as both employ the same behavior.

Our goal is to compute σ and τ from the perspective of Pool 1, given this strategy regime. Each round of the game has two state variables. The first state variable, i, will refer to how far behind the compliant chain is. Then $i = -1$ denotes the fact that the compliant chain has won the round, while $i = 0$ means the NC chain has been equalized by the compliant chain, so the first-seen rule favors the NC chain. At block k, C will concede the transaction. The second state variable, s, will refer to the number of block rewards Pool 1 has contributed to the NC chain.

A third variable, which is determined when the game begins and does not change after the first block is mined, will be the amount that Pool 1 has won by mining the offending transaction, taking values in $\{0, \lambda\}$ (either they mined the block or some other NC mined the block). This does not need to be dragged through computations along the game, but is necessary information when the game ends in favor of the NC chain.

While the game begins when the objectionable transaction is posted, the round begins (as with all subsequent rounds) when the transaction is mined and the longest chain

becomes non-compliant. The block will have been mined by Pool 1, Pool 2, or NC. With probability $\varepsilon_1/(1-p)$ the round begins in state $(1,1)$ and with $1-\varepsilon_1/(1-p)$ probability the game begins in state $(1,0)$. We consider these cases to begin computing the expected value of L_1:

$$\mathbb{E}[L_1] = \frac{\varepsilon_1}{1-p}\mathbb{E}\big[L_1 \mid (1,1)\big] + \left(1 - \frac{\varepsilon_1}{1-p}\right)\mathbb{E}\big[L_1 \mid (1,0)\big]. \tag{6.5}$$

Using conditional expectations,

$$\mathbb{E}\big[L_1 \mid (x,s)\big] = \left\{ \begin{array}{l} p\mathbb{E}[L_1 \mid (x-1,s)] \\ \quad + (q+\varepsilon_2)\mathbb{E}[L_1 \mid (x+1,s)] \\ \quad + \varepsilon_1\mathbb{E}[L_1 \mid (x+1,s+1)] \end{array} \right\}. \tag{6.6}$$

This incorporates the three possibilities; the next block is won by C, NC/Pool 2, or Pool 1. We use a fact that can be concluded from an argument similar to the one leading to Claim 5.1:

$$\mathbb{E}\big[L_1 \mid (x,s+1)\big] = \mathbb{E}\big[L_1 \mid (x,s)\big] + w(x). \tag{6.7}$$

Letting

$$e_1(x) = \mathbb{E}\big[L_1 \mid (x,0)\big],$$

we have from (6.6) and (6.7) with $s = 0$

$$\begin{aligned} e_1(x) &= pe_1(x-1) + (q+\varepsilon_2)e_1(x+1) + \varepsilon_1\big(e_1(x+1) + w(x+1)\big) \\ &= pe_1(x-1) + (1-p)e_1(x+1) + \varepsilon_1 w(x+1). \end{aligned}$$

Because for the entirety of the round, C is playing catch-up, the probability that C wins w can be computed exactly as in §5.1.3:

$$w(x) = c_1\left(\frac{p}{1-p}\right)^{x+1} + c_2,$$

where c_1, c_2 are determined by (5.13). So we have the equation

$$e_1(x) - pe_1(x-1) - (1-p)e_1(x+1) = \varepsilon_1\left(c_1\left(\frac{p}{1-p}\right)^{x+1} + c_2\right), \tag{6.8}$$

which is quite similar to equation (5.16).

As before,

$$y_1 = -\varepsilon_1\frac{c_2}{1-2p}x,$$

$$y_2 = \frac{\varepsilon_1 c_1 x}{1-2p}\left(\frac{p}{1-p}\right)^{x+1}$$

will solve the equations

$$\begin{aligned} y(x) - py(x-1) - (1-p)y(x+1) &= \varepsilon_1 c_2, \\ y(x) - py(x-1) - (1-p)y(x+1) &= \varepsilon_1 c_1 \left(\frac{p}{1-p}\right)^{x+1}. \end{aligned}$$

So the general solution is

$$y(x) = \tilde{c}_1\left(\frac{p}{1-p}\right)^x + \tilde{c}_2 - \varepsilon_1 \frac{c_2}{1-2p}x + \frac{\varepsilon_1 c_1 x}{1-2p}\left(\frac{p}{1-p}\right)^{x+1}. \tag{6.9}$$

We can solve this with boundary conditions

$$\begin{aligned} e_1(-1) &= 0, \\ e_1(k) &= 0. \end{aligned}$$

Again recall why the zero boundary conditions are there: Because we are solving for $s = 0$ in the second state variable, there will be nothing gained nor lost by the mining pool.

Solving,

$$\begin{aligned} \tilde{c}_1\left(\frac{p}{1-p}\right)^{-1} + \tilde{c}_2 + \varepsilon_1 \frac{c_2}{1-2p} - \frac{\varepsilon_1 c_1}{1-2p} &= 0, \\ \tilde{c}_1\left(\frac{p}{1-p}\right)^{k} + \tilde{c}_2 - \varepsilon_1 \frac{c_2}{1-2p}k + \frac{\varepsilon_1 c_1}{1-2p}k\left(\frac{p}{1-p}\right)^{k+1} &= 0, \end{aligned}$$

or

$$\begin{pmatrix} m^{-1} & 1 \\ m^k & 1 \end{pmatrix}\begin{pmatrix} \tilde{c}_1 \\ \tilde{c}_2 \end{pmatrix} = \frac{\varepsilon_1}{1-2p}\begin{pmatrix} 1 & -1 \\ -km^{k+1} & k \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}.$$

As before, we get

$$\begin{pmatrix} \tilde{c}_1 \\ \tilde{c}_2 \end{pmatrix} = \frac{m}{1-m^{k+1}}\frac{\varepsilon_1}{1-2p}\begin{pmatrix} 1 & -1 \\ -m^k & m^{-1} \end{pmatrix}\begin{pmatrix} 1 & -1 \\ -km^{k+1} & k \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}. \tag{6.10}$$

With these values evaluated we have the solution:

$$e_1(x) = \tilde{c}_1 m^x + \tilde{c}_2 - \varepsilon_1 \frac{c_2}{(1-2p)}x + \varepsilon_1 \frac{c_1}{1-2p}xm^{x+1}.$$

Now we are interested in the values

$$\mathbb{E}[L_1 \mid (1,0)] = e_1(1) = \tilde{c}_1 m + \tilde{c}_2 - \varepsilon_1 \frac{c_2}{(1-2p)} + \varepsilon_1 \frac{c_1}{1-2p} m^2,$$
$$\mathbb{E}[L_1 \mid (1,1)] = e_1(1) + w(1). \tag{6.11}$$

We have determined that the expected number of blocks that are lost in a given round, recalling (6.5), is

$$\begin{aligned}\mathbb{E}[L_1] &= \frac{\varepsilon_1}{1-p}\big(e_1(1) + w(1)\big) + \left(1 - \frac{\varepsilon_1}{1-p}\right) e_1(1) \\ &= e_1(1) + \frac{\varepsilon_1}{1-p} w(1).\end{aligned}$$

We are not done yet – we wanted to know the expected number of blocks that are lost, given the fact that C won the round. To compute this use

$$\mathbb{E}[L_1] = w(1)\mathbb{E}[L_1 \mid \text{C wins}] + \big(1 - w(1)\big)\mathbb{E}[L_1 \mid \text{C loses}].$$

Because

$$\mathbb{E}[L_1 \mid \text{C loses}] = 0,$$

we can determine that

$$\begin{aligned}\mathbb{E}[L_1 \mid \text{C wins}] &= \frac{\mathbb{E}[L_1]}{w(1)} \\ &= \frac{e_1(1)}{w(1)} + \frac{\varepsilon_1}{1-p}.\end{aligned} \tag{6.12}$$

Thus we determine

$$\tau = \frac{e_1(1)}{w(1)} + \frac{\varepsilon_1}{1-p}. \tag{6.13}$$

Now σ is easier; it is the expected share of the additional reward. This is easily seen to be

$$\sigma = \frac{\varepsilon_1}{1-p}\lambda$$

as once the round starts the odds of winning do not depend on which pool mined the offending block. So the payoff to Pool 1 is

$$\begin{aligned}&\frac{\varepsilon_1}{1-p}\lambda - \left(\frac{e_1(1)}{w(1)} + \frac{\varepsilon_1}{1-p}\right)\frac{w(1)}{1-w(1)} \\ &\quad = \frac{\varepsilon_1}{1-p}\left(\lambda - \frac{w(1)}{1-w(1)}\right) - \frac{e(1)}{1-w(1)},\end{aligned} \tag{6.14}$$

where $e_1(1)$ is determined by (6.11) and (6.10).

Summary
To summarize:
1. Solve c_1, c_2 via (5.13).
2. Solve $\tilde{c}_1, \tilde{c}_2$ via (6.10).
3. Solve $e_1(1)$ using (6.11).
4. Note that $w(1) = c_1 m + c_2$.
5. Harvest the result in (6.14). This is $\mathbb{E}[\lambda W_1 - L_1]$ in the (non-compliant, non-compliant) strategy regime.

The computation for Pool 2 is identical. One only needs to modify (3) to incorporate the different value ε_2.

6.1.4 Case 2: Pool 1 is non-compliant and Pool 2 is mildly compliant

We will begin by attempting to compute $w(x)$. However, there is a wrinkle that prohibits us from continuing to use exactly the same calculus leading up to (5.13). These finite difference equations to be solved are "broken" by the fact that the strategy and transition probabilities change at intermediate states. So we need to perform some slightly more involved computations in order to solve the problem.

6.1.4.1 Broken difference equations

We begin with the most basic computation, the probability $w(x)$ that C wins the round, given we are at state x. The equations are different from previously. In particular, letting

$$p^* = p + \varepsilon_2,$$

we have

$$w(0) = p^* w(-1) + (1 - p^*) w(1). \tag{6.15}$$

This encodes the fact that when $i = 0$, Pool 2's hashrate of ε_2 is joining the censors (hence p^* and not p). However, for integers $x \geq 1$, Pool 2 returns to mine on the true chaintip, and we have

$$w(x) = p w(x - 1) + (1 - p) w(x + 1). \tag{6.16}$$

We solve the broken difference equation using somewhat of a cheat: Look at equation (6.16). This holds for $x > 0$ and only excludes the point 0, so we may think of (6.15) as a single relation that needs to be satisfied and solve the problem on $[0, k]$ considering this single relation and one degree of flexibility built into the boundary value problem. Note that this works when there is only one relation in the set of relations (6.15).

To begin, because $w(-1) = 1$, from (6.15) we have

$$w(0) = p^* + (1 - p^*)w(1). \tag{6.17}$$

Now if we solve the boundary value problem for equation (6.16) with

$$\begin{aligned} w(0) &= b, \\ w(k) &= 0 \end{aligned}$$

(for b to be determined), then necessarily from (6.17) we see that

$$w(1) = \frac{b - p^*}{1 - p^*}$$

(we have skipped the step of solving the first equation (6.15) as an equation; knowing $w(-1) = 1$, specifying $w(0) = b$, we know what $w(1)$ must be). So now, we may solve equation (6.16) with three values,

$$\begin{aligned} w(0) &= b, \\ w(1) &= \frac{b - p^*}{1 - p^*}, \\ w(k) &= 0. \end{aligned}$$

This extra parameter b allows us room to put in the third condition, and we expect to get a unique solution.

Equation (6.16) is the same equation we have seen before, with general solution given by

$$w(x) = c_1 m^x + c_2.$$

Thus the boundary conditions above become

$$\begin{aligned} c_1 + c_2 &= b, \\ c_1 m + c_2 &= \frac{b - p^*}{1 - p^*}, \\ c_1 m^k + c_2 &= 0. \end{aligned}$$

As a matrix,

$$\begin{pmatrix} 1 & 1 \\ m & 1 \\ m^k & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} b \\ \frac{b-p^*}{(1-p^*)} \\ 0 \end{pmatrix}. \tag{6.18}$$

Now we augment the matrix in order to solve the problem completely using linear algebra. Using that

$$\begin{pmatrix} b \\ \frac{b-p^*}{1-p^*} \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{1-p^*} \\ 0 \end{pmatrix} b + \begin{pmatrix} 0 \\ \frac{-p^*}{1-p^*} \\ 0 \end{pmatrix},$$

we see that (6.18) is equivalent to

$$\begin{pmatrix} 1 & 1 \\ m & 1 \\ m^k & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} - \begin{pmatrix} 1 \\ \frac{1}{1-p^*} \\ 0 \end{pmatrix} b = \begin{pmatrix} 0 \\ \frac{-p^*}{1-p^*} \\ 0 \end{pmatrix}$$

or

$$\begin{pmatrix} 1 & 1 & -1 \\ m & 1 & \frac{-1}{1-p^*} \\ m^k & 1 & 0 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ b \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{-p^*}{1-p^*} \\ 0 \end{pmatrix}.$$

Now, in order to solve for c_1 and c_2 (and b incidentally) we need to invert this 3×3 matrix,

$$\begin{pmatrix} c_1 \\ c_2 \\ b \end{pmatrix} = \begin{pmatrix} 1 & 1 & -1 \\ m & 1 & \frac{-1}{1-p^*} \\ m^k & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ \frac{-p^*}{1-p^*} \\ 0 \end{pmatrix}. \tag{6.19}$$

Now that we have c_1 and c_2, we can compute our new function $w(x)$, now given as a multi-part function:

$$\begin{aligned} w(-1) &= 1, \\ w(x) &= c_1 m^x + c_2 \quad \text{for } x \geq 0. \end{aligned}$$

Remark 6.2. If we had cutoff at, say, $i = 2$, the computation would have been more complicated. Another method would be to solve both equations, but with "buckled" boundary conditions that overlap at two points and a free parameter (such a b above) specified for one of the boundary conditions.

6.1.4.2 Expected value for Pool 1

This same strategy works to compute expected values. To begin, we have

$$\mathbb{E}[L_1] = \frac{\varepsilon_1}{q+\varepsilon_1}\mathbb{E}[L_1 \mid (1,1)] + \frac{q}{q+\varepsilon_1}\mathbb{E}[L_1 \mid (1,0)]. \tag{6.20}$$

But then, using $e_1(x,s)$ for short-hand we have a single relation describing the transitions at state $(0,s)$:

$$e_1(0,s) = p^* e_1(-1,s) + \varepsilon_1 e_1(1,s+1) + qe(1,s).$$

As in (6.7),

$$e_1(1,s+1) = e_1(1,s) + w(1), \tag{6.21}$$

so

$$e_1(0,s) = p^* e_1(-1,s) + \varepsilon_1\big(e_1(1,s) + w(1)\big) + qe_1(1,s),$$

or

$$e_1(0,s) - p^* e_1(-1,s) - (1-p^*)e_1(1,s) = \varepsilon_1 w(1). \tag{6.22}$$

As above when we solved the broken equation for $w(x)$, this relation will be plugged in directly to create a three-parameter boundary value problem when solving for $e_1(x)$.

For $x \geq 1$,

$$e_1(x,s) = pe(x-1,s) + \varepsilon_1 e_1(x+1,s+1) + (q+\varepsilon_2)e_1(x+1,s),$$

which gives a familiar equation:

$$e_1(x,s) - e_1(x-1,s) - (1-p)e_1(x+1,s) = \varepsilon_1 w(x+1). \tag{6.23}$$

The general strategy for solving this type of broken system of non-homogeneous equations is similar to the strategy for solving homogeneous equations. We will set up the general solution for (6.23) and then use (6.22) to set up a third condition.

Solving (6.8) as before to get the general solution (6.9),

$$y(x) = \tilde{c}_1\left(\frac{p}{1-p}\right)^x + \tilde{c}_2 - \varepsilon_1\frac{c_2}{1-2p}x + \frac{\varepsilon_1 c_1 x}{1-2p}\left(\frac{p}{1-p}\right)^{x+1}, \tag{6.24}$$

where c_1, c_2 are obtained by (6.19). Set boundary conditions

$$\begin{aligned} e_1(0) &= b_1, \\ e_1(k) &= 0 \end{aligned}$$

and then bring in (6.22):

$$e_1(1) = \frac{b_1 - \varepsilon_1 w(1)}{1-p^*}. \tag{6.25}$$

We are trying to solve for $\tilde{c}_1, \tilde{c}_2$, so we plug the values for e_1 in the three previous expressions to get

$$\begin{aligned}
\tilde{c}_1 + \tilde{c}_2 &= b_1,\\
\tilde{c}_1 m + \tilde{c}_2 - \varepsilon_1 \frac{c_2}{1-2p} + \frac{\varepsilon_1 c_1}{1-2p} m^2 &= \frac{b_1 - \varepsilon_1 w(1)}{1-p^*},\\
\tilde{c}_1 m^k + \tilde{c}_2 - \varepsilon_1 \frac{c_2}{1-2p} k + \frac{\varepsilon_1 c_1 k}{1-2p} m^{k+1} &= 0,
\end{aligned}$$

which tidies up into matrix notation as

$$\begin{aligned}
\begin{pmatrix} 1 & 1 & -1 \\ m & 1 & -\frac{1}{1-p^*} \\ m^k & 1 & 0 \end{pmatrix} \begin{pmatrix} \tilde{c}_1 \\ \tilde{c}_2 \\ b_1 \end{pmatrix} &= \begin{pmatrix} 0 \\ \frac{-\varepsilon_1 w(1)}{1-p^*} + \varepsilon_1 \frac{c_2}{1-2p} - \frac{\varepsilon_1 c_1}{1-2p} m^2 \\ \varepsilon_1 \frac{c_2}{1-2p} k - \frac{\varepsilon_1 c_1 k}{1-2p} m^{k+1} \end{pmatrix} \\
&= \varepsilon_1 \begin{pmatrix} 0 & 0 & 0 \\ \frac{-w(1)}{1-p^*} & -\frac{m^2}{1-2p} & \frac{1}{1-2p} \\ 0 & -\frac{k}{1-2p} m^{k+1} & \frac{k}{1-2p} \end{pmatrix} \begin{pmatrix} 1 \\ c_1 \\ c_2 \end{pmatrix}. \qquad (6.26)
\end{aligned}$$

We can solve for the coefficients $\tilde{c}_1, \tilde{c}_2$ using linear algebra. Once we have done this, we plug back into (6.20) and (6.21) and compute

$$\begin{aligned}
\mathbb{E}[L_1] &= \frac{\varepsilon_1}{q+\varepsilon_1}\big(e_1(1) + w(1)\big) + \frac{q}{q+\varepsilon_1} e(1)\\
&= e_1(1) + \frac{\varepsilon_1}{q+\varepsilon_1} w(1).
\end{aligned}$$

As before (equations (6.12), (6.13)), we can then determine that

$$\begin{aligned}
\tau &= \frac{e_1(1)}{w(1)} + \frac{\varepsilon_1}{q+\varepsilon_1},\\
\sigma &= \frac{\varepsilon_1}{q+\varepsilon_1}\lambda.
\end{aligned}$$

Once the round starts, the odds of winning do not depend on which pool mined the offending block. So the payoff to Pool 1 is

$$\begin{aligned}
&\frac{\varepsilon_1}{q+\varepsilon_1}\lambda - \left(\frac{e_1(1)}{w(1)} + \frac{\varepsilon_1}{q+\varepsilon_1}\right)\frac{w(1)}{1-w(1)}\\
&\quad = \frac{\varepsilon_1}{q+\varepsilon_1}\left(\lambda - \frac{w(1)}{1-w(1)}\right) - \frac{e_1(1)}{1-w(1)}. \qquad (6.27)
\end{aligned}$$

This gives the expected value for Pool 1 in the second case.

Summary

1. Solve for w, that is, find coefficients c_1, c_2 using (6.19).
2. Solve for $\tilde{c}_1, \tilde{c}_2, b_1$ using (6.26) and $w(1) = c_1 m + c_2$.

3. Solve for $e_1(1)$ using (6.25).
4. Harvest $\mathbb{E}[\lambda W_1 - L_1]$ using (6.27).

We note that because we are using b_1 to solve for the value $e_1(1)$, we end up not needing the information given by $\tilde{c}_1$ and $\tilde{c}_2$ in this particular computation.

6.1.4.3 Expected value for Pool 2

We have already computed $w(x)$ in this strategy regime. In setting up the computation of the outcome for Pool 2, we now let the second state variable denote the number of blocks that Pool 2 has mined. Note that because Pool 2 is mining with the censorial pool only when the chains are of equal length, Pool 2 has no risk of censorial blocks being lost. The risk to Pool 2 is realized only if Pool 2 adds to an NC chain that is already one block ahead and this chain ends up losing (the given strategy allows for the unlikely but still possible event that the chain will mine a block on the NC chain and then end up breaking a tie in favor of C, making an uncle of their own block, so the strategy would probably be tweaked in practice).

Because the pool is not interested in mining the objectionable transaction itself, we start with

$$\mathbb{E}[L_2] = \mathbb{E}\big[L_2 \mid (1,0)\big] \tag{6.28}$$

as the initial state necessarily involves either Pool 1 or NC having just mined the block. Now

$$\mathbb{E}\big[L_2 \mid (0,s)\big] = (p+\varepsilon_2)\mathbb{E}\big[L_2 \mid (-1,s)\big] + (1-p-\varepsilon_2)\mathbb{E}\big[L_2 \mid (1,s)\big],$$

and for $x \geq 1$,

$$\mathbb{E}\big[L_2 \mid (x,s)\big] = p\mathbb{E}\big[L_2 \mid (x-1,s)\big] + (1-p-\varepsilon_2)\mathbb{E}\big[L_2 \mid (x+1,s)\big] + \varepsilon_2\mathbb{E}\big[L_2 \mid (x+1,s+1)\big]. \tag{6.29}$$

Also, as before,

$$\mathbb{E}\big[L_2 \mid (x,s+1)\big] = \mathbb{E}\big[L_2 \mid (x,s)\big] + w(x).$$

Using $e_2(x) = \mathbb{E}[L_2 \mid (x,0)]$, (6.29) becomes

$$e_2(x) - pe_2(x-1) - (1-p)e_2(x+1) = \varepsilon_2 w(x+1) \text{ for } x > 0.$$

The function $w(x)$ has been computed with coefficients determined by (6.19). The analysis to determine $e_2(x)$ is very similar to the analysis line following (6.24). With general solution

$$y(x) = \tilde{c}_1\left(\frac{p}{1-p}\right)^x + \tilde{c}_2 - \varepsilon_2\frac{c_2}{1-2p}x + \frac{\varepsilon_2 c_1 x}{1-2p}\left(\frac{p}{1-p}\right)^{x+1}, \tag{6.30}$$

we solve for boundary conditions

$$\begin{aligned} e_2(0) &= b_2, \\ e_2(k) &= 0 \end{aligned} \tag{6.31}$$

and the additional condition with free parameter b_2

$$e_2(1) = \frac{b_2}{1-p^*}. \tag{6.32}$$

This follows from (6.31) and

$$e_2(-1) = 0$$

and

$$e_2(0) - p^* e_2(-1) - (1-p^*)e_2(1) = 0, \tag{6.33}$$

which differs from (6.22) because there is no $w(1)$ term on the right-hand side.

We are trying to solve for $\tilde{c}_1, \tilde{c}_2$, so we plug the boundary values in and convert to a matrix:

$$\begin{aligned} \begin{pmatrix} 1 & 1 & -1 \\ m & 1 & -\frac{1}{1-p^*} \\ m^k & 1 & 0 \end{pmatrix} \begin{pmatrix} \tilde{c}_1 \\ \tilde{c}_2 \\ b_2 \end{pmatrix} &= \begin{pmatrix} 0 \\ \varepsilon_2 \frac{c_2}{1-2p} - \frac{\varepsilon_2 c_1}{1-2p} m^2 \\ \varepsilon_2 \frac{c_2}{1-2p} k - \frac{\varepsilon_2 c_1 k}{1-2p} m^{k+1} \end{pmatrix} \\ &= \frac{\varepsilon_2}{1-2p} \begin{pmatrix} 0 & 0 \\ -m^2 & 1 \\ -km^{k+1} & k \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}. \end{aligned} \tag{6.34}$$

Inverting the first matrix, we can solve for the coefficients $\tilde{c}_1$, $\tilde{c}_2$, and b_2. Once we have done this, we plug back into

$$\mathbb{E}[L_2] = e_2(1) = \frac{b_2}{1-p^*}. \tag{6.35}$$

As before we can then determine that

$$\begin{aligned} \tau &= \frac{e_2(1)}{w(1)}, \\ \sigma &= 0. \end{aligned}$$

So the total expectation for Pool 2 is

$$-\frac{e_2(1)}{1-w(1)} = -\frac{b_2}{(1-w(1))(1-p^*)}. \tag{6.36}$$

Summary

1. Solve for w, that is, find coefficients c_1, c_2 using (6.19).
2. Solve for $\tilde{c}_1, \tilde{c}_2, b_2$ using (6.34) and $w(1) = c_1 m + c_2$.
3. Solve for $e_2(1)$ using (6.32).
4. Harvest $\mathbb{E}[\lambda W_2 - L_2]$ using (6.36).

6.1.5 Case 3: both pools are mildly compliant

This will be almost identical to the computation in §6.1.4.3. This time we take

$$p^* = p + \varepsilon_1 + \varepsilon_2.$$

Because neither pool is interested in mining the objectionable transaction itself, we start with

$$\mathbb{E}[L_i] = \mathbb{E}\big[L_i \mid (1,0)\big]$$

as the initial state is that only NC has mined the block. Now we have

$$\mathbb{E}\big[L_i \mid (0,s)\big] = p^* \mathbb{E}\big[L_i \mid (-1,s)\big] + (1-p^*)\mathbb{E}\big[L_i \mid (1,s)\big]$$

for $x = 0$ and

$$\mathbb{E}\big[L_i \mid (x,s)\big] = p\mathbb{E}\big[L_i \mid (x-1,s)\big] + (1-p-\varepsilon_i)\mathbb{E}\big[L_i \mid (x+1,s)\big] + \varepsilon_i \mathbb{E}\big[L_i \mid (x+1,s+1)\big]$$

for $x \geq 1$. Also, as before,

$$\mathbb{E}\big[L_i \mid (x,s+1)\big] = \mathbb{E}\big[L_i \mid (x,s)\big] + w(x),$$

leading to

$$e_i(x) - pe_i(x-1) - (1-p)e_i(x+1) = \varepsilon_i w(x+1) \quad \text{for } x \geq 1.$$

The function $w(x)$ is determined from coefficients given by (6.19). Setting boundary conditions

$$\begin{aligned} e_i(0) &= b_i^*, \\ e_i(1) &= \frac{b_i^*}{1-p^*}, \\ e_i(k) &= 0 \end{aligned} \tag{6.37}$$

gives us a matrix equation:

$$\begin{pmatrix} 1 & 1 & -1 \\ m & 1 & -\frac{1}{1-p^*} \\ m^k & 1 & 0 \end{pmatrix} \begin{pmatrix} \tilde{c}_1 \\ \tilde{c}_2 \\ b_i^* \end{pmatrix} = \varepsilon_i \begin{pmatrix} 0 & 0 \\ -\frac{m^2}{1-2p} & \frac{1}{1-2p} \\ -\frac{k}{1-2p} m^{k+1} & \frac{k}{1-2p} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}. \tag{6.38}$$

Inverting the matrix, we can solve for the coefficients $\tilde{c}_1, \tilde{c}_2$. Once we have done this,

$$\mathbb{E}[L_i] = e_i(1). \tag{6.39}$$

So the total expectation for Pool 2 is

$$-\frac{e_i(1)}{1-w(1)} \tag{6.40}$$

$$= -\frac{b_i^*}{(1-w(1))(1-p^*)}. \tag{6.41}$$

Inspecting (6.38) we see that

$$\frac{b_1^*}{b_2^*} = \frac{\varepsilon_1}{\varepsilon_2}.$$

Summary

1. Solve for w, that is, find coefficients c_1, c_2 using (6.19).
2. For either ε_1 or ε_2, solve for $\tilde{c}_1$, $\tilde{c}_2$, and b using (6.38) and $w(1) = c_1 m + c_2$.
3. Solve for $e_i(1)$ using (6.37).
4. Harvest $\mathbb{E}[\lambda W_2 - L_2]$ using (6.40).

6.1.6 Analysis: low-fee regime

We begin by looking at situations in which the fee paid by the censored party is relatively small. As of September 2022, the block reward is about \$125,000. So if we assume the censored party is willing to part with \$125 in order to process the transaction, this will amount to $\lambda = 0.001$.

6.1.6.1 Normalized tables

We will use "normalized tables," which divide the expectations of a pool by the hashrate of that same pool. Re-normalization of any agent's payoff will not affect strategic considerations, because it does not change preferential ordering. Dominant strategies will remain dominant, and so forth. Agents are attempting to maximize their own outcomes, not their outcomes relative to someone else's outcomes. Note that the hashing fraction of a pool represents the expected block rewards per 10 minutes of mining. So the payoff number given in the normalized table is given in units of expected payoff for 10 minutes of mining for that given pool.

To illustrate this, we start with $p = 0.2$, $\varepsilon_1 = 0.15$, $\varepsilon_2 = 0.1$. A censoring pool has 20 % of hashing power, and two other rational pools with hashing powers of 15 % and 10 %

are deciding between non-compliant and mildly compliant strategies. We start with the unnormalized table with $\lambda = 0.001$ and $k = 4$:

	Non-compliant	**Mildly compliant**
Non-compliant	(−0.01944, −0.01296)	(−0.03154, −0.00416)
Mildly compliant	(−0.00719, −0.0256)	(−0.00897, −0.00598)

Unnormalized payoff: $p = 0.2, \varepsilon_1 = 0.15, \varepsilon_2 = 0.1, \lambda = 0.001, k = 4$.

For comparison, we also show the normalized payoff table:

	Non-compliant	**Mildly compliant**
Non-compliant	(−0.1296, −0.1296)	(−0.2103, −0.0416)
Mildly compliant	(−0.0479, −0.256)	(−0.0598, −0.0598)

Normalized payoff: $p = 0.2, \varepsilon_1 = 0.15, \varepsilon_2 = 0.1, \lambda = 0.001, k = 4$.

Observe this normalized table has some symmetry along the diagonal: It represents that for a censor with censoring power 20 % and a transaction only offering 0.1 % of a block reward as compensation, if both miners use a non-compliant strategy, they are each expected to lose about 13 % of their expected reward over 10 minutes of typical mining (the game itself could last several blocks). By comparison, if both mining pools adopt a mildly compliant strategy they expect to lose about 6 % of their expectations over 10 minutes.

We observe quite clearly the type of game this represents. Both players will play their dominant strategy, and this will maximize the payoff to both players collectively. This is called a *deadlock game*. A deadlock game is a game in which each player has a clearly dominant strategy, and this is mutually beneficial. Typically such games are less interesting, because players choose the strategy that benefits others represented in the payoff table out of their own self-interest. Of course, the party whose interest is not represented by this table is the party sending the transaction. However, once they have sent the transaction and offered a fee, they have no agency in the mining game.

Both pools are motivated to use a mildly compliant strategy, at least when given the choice between mildly compliant and non-compliant. The strategy space is quite large, so we cannot conclude that this strategy is dominant among all possible strategies; it is only dominant when compared against a non-compliant strategy. We should be careful to put this into perspective offered by the normalized table; a total of 7 % of a pool's block reward is about 4 minutes of mining time.

A pool of size 20 % may be considered a large censoring pool, as it represents already 40 % of the way towards full control of the network. How effective is censorship with lower combinations of hashpower? Reducing this censor's power to 5 % shows a marked difference:

	Non-compliant	Mildly compliant
Non-compliant	(−0.00219, −0.00219)	(−0.0096, −0.0005)
Mildly compliant	(−0.00067, −0.0139)	(−0.0010, −0.0010)

Normalized payoff: $p = 0.05, \varepsilon_1 = 0.15, \varepsilon_2 = 0.1, \lambda = 0.001, k = 4$.

Note that while the dynamics of the game are still essentially the same, for low hashrate of the censor the motivations towards rational pools may be approaching indifference: lose 0.2 % of your expected rewards for 10 minutes of mining versus lose 0.1 %. This is equivalent to a difference of about 6 seconds of mining.

What if we consider larger rational pools?

	Non-compliant	Mildly compliant
Non-compliant	(−0.00219, −0.00219)	(−0.00287, −0.000203)
Mildly compliant	(−0.00182, −0.0691)	(−0.00185, −0.00185)

Normalized payoff: $p = 0.05, \varepsilon_1 = 0.5, \varepsilon_2 = 0.01, \lambda = 0.001, k = 4$.

Still, the incentive to comply is still rather small, approaching indifference.

Now what about when the C pool is larger, say 35 %? First we consider small-sized rational pools:

	Non-compliant	Mildly compliant
Non-compliant	(−1.204, −1.204)	(−1.204, −0.423)
Mildly compliant	(−0.423, −1.204)	(−0.423, −0.423)

Normalized payoff: $p = 0.35, \varepsilon_1 = 0.0001, \varepsilon_2 = 0.0001, \lambda = 0.001, k = 4$.

Perhaps not surprisingly, when the rational pools are small, the payoff tables are nearly independent of the other pool's behavior (they actually are different, but the difference is smaller than the precision offered in the tables). For either pool there is a stronger incentive (about 8 minutes of mining) to switch to a mildly compliant strategy.

For comparison:

	Non-compliant	Mildly compliant
Non-compliant	(−1.204, −1.204)	(−1.612, −0.5065)
Mildly compliant	(−0.5795, −2.126)	(−0.6438, −0.6438)

Normalized payoff: $p = 0.35, \varepsilon_1 = 0.2, \varepsilon_2 = 0.1, \lambda = 0.001, k = 4$.

Unsurprisingly, the incentive becomes more certain as the censorial pool approaches 50 %:

	Non-compliant	Mildly compliant
Non-compliant	(−5.565, −5.565)	(−6.196, −2.321)
Mildly compliant	(−2.321, −6.196)	(−2.416, −2.416)

Normalized payoff: $p = 0.47, \varepsilon_1 = 0.05, \varepsilon_2 = 0.05, \lambda = 0.001, k = 4$.

Here a decision to resist the censors will result in an additional expected loss of over 20 minutes of mining. This may still seem small. When the censorial pool extends k (which will make more sense as $p \to 0.5$) there is a significant increase in expected loss:

	Non-compliant	Mildly compliant
Non-compliant	(−29.224, −29.224)	(−30.952, −21.571)
Mildly compliant	(−21.571, −30.952)	(−22.096, −22.096)

Normalized payoff: $p = 0.47, \varepsilon_1 = 0.05, \varepsilon_2 = 0.05, \lambda = 0.001, k = 10$.

6.1.7 Higher-fee regime

The game changes if the party sending the offending transaction is willing to pay. In the following, we take

$$\lambda = 0.16,$$

which represents one bitcoin when the block reward is 6.25 bitcoins.

	Non-compliant	Mildly compliant
Non-compliant	(0.06912, 0.06912)	(0.04383, −0.03525)
Mildly compliant	(−0.03525, 0.04383)	(−0.04167, −0.04167)

Normalized payoff: $p = 0.2, \varepsilon_1 = 0.05, \varepsilon_2 = 0.05, \lambda = 0.16, k = 4$.

The tables have turned. The non-compliant strategy is now dominant, at least when $p = 0.2$. Increasing to $p = 0.25$, we have:

	Non-compliant	Mildly compliant
Non-compliant	(−0.07725, −0.07725)	(−0.1306, −0.09205)
Mildly compliant	(−0.09205, −0.1306)	(−0.1052, −0.1052)

Normalized payoff: $p = 0.25, \varepsilon_1 = 0.05, \varepsilon_2 = 0.05, \lambda = 0.16, k = 4$.

Note that this game has no dominant strategies available, and it has two Nash equilibria, similar to the stag hunt.

6.1.8 Conclusion

In terms of pure game-theoretical considerations, it is possible for a pool with censorial objectives to change the incentives landscape so that rational pools are motivated to join the censors. However, from the tables above, if the pool controls a smaller portion, say 5 % of the hashrate, the motivations are likely to not come into play, unless there are extremely low profit margins for the miners. A censorial pool cannot expect to win all their battles, but the goal would be to cause enough losses over time so the rational miners would join them in increasing numbers. The effect is to grind down the non-compliant pools making non-compliant mining less profitable on average.

There are many other possible strategies that can be explored, and this becomes much more interesting when profitability of the miners is taken into consideration. We are not considering the possibility that non-compliant pools will punish those that join compliant efforts by breaking consensus protocols to mine non-compliant blocks, even at a loss.

Neither are we considering competition for blockspace. A computation (more favorable to the censors) would subtract from λ the average fee obtainable for the blockspace, according to the market rates. In 2022, this might not be a large difference, but as the block reward decreases over the next 40 years, this modified computation may become more significant.