Foundations of Blockchains
Lecture #10: Selfish Mining
(IN PROGRESS)[*]

Tim Roughgarden[†]

1 The Upshot

1. foo

2 Block Rewards

2.1 Blockchain Protocols with a Native Currency

Thus far, we've been studying possibility and impossibility results in consensus, the problem of keeping a bunch of computers in sync despite network delays and misbehaving nodes. In classical applications of consensus (e.g., a big company like IBM replicating a database to achieve very high uptime), there's no reason to think about the incentives of running a node (the company just runs them all as part of its business). But in the vision of permissionless consensus, nodes are arbitrary individuals, free to come and go as they please. Why should anyone bother to help run the protocol? This question becomes particularly acute with Nakamoto consensus (Lecture 9), in which nodes are expected to exert significant computational effort to solve hard cryptopuzzles.

A second question you might be wondering about is: Why we haven't mentioned cryptocurrencies? Aren't they the whole point of blockchain technology? While the stories of blockchains and cryptocurrencies have been tightly intertwined thus far (and may continue to be for quite awhile), fundamentally, the answer is no. As we've seen, it's perfectly possible in principle to build consensus protocols (including permissionless ones) that don't come with a cryptocurrency.

Starting with this lecture (and through Lecture 13), we'll focus specifically on blockchain protocols with a native currency—meaning (digital) coins that are minted, destroyed, and

[*]©2021–2023, Tim Roughgarden.

[†]Email: `tim.roughgarden@gmail.com`. The preparation of these notes was partially supported by a seed grant from the Columbia-IBM Center for Blockchain and Data Transparency. These notes are a work in progress; corrections and feedback are welcome.

tracked by the protocol itself—and the new challenges and opportunities that then arise. We'll have a lot to say about cryptocurrencies, but for this lecture we'll focus quite narrowly on using a native currency to answer the first question above: Why should anyone bother to participate in a permissionless consensus protocol?[1]

2.2 Incentivizing Block Production Through Block Rewards

One convenient way to incentivize nodes to participate in a blockchain protocol is through *block rewards*. For example, in Nakamoto consensus, you could imagine giving an economically meaningful reward (enough to cover depreciation, electricity, and opportunity costs) to a node whenever it successfully solves a cryptopuzzle and proposes a block that winds up on the longest chain.

Where might a blockchain protocol get an "economically meaningful reward" from? The easiest solution is for the protocol to print the money itself (necessarily, in the only currency that it controls) and dole out block rewards denominated in the native cryptocurrency.[2] This is precisely what happens in the Bitcoin protocol, for example—at the time of this writing, the miner of a Bitcoin block that gets included in the longest chain gets rewarded with 6.25 BTC (which, at the time of this writing, is over $100K).[3]

Introducing incentives into any system (be it in web3, web2, or the real world) can lead to counterintuitive and undesirable outcomes. Sure, block rewards encourage nodes to produce

[1]A brief look ahead: In Lecture 11 we'll consider transaction fees, paid by end users in return for the execution of their transaction by a blockchain protocol. There are advantages (and it is common practice) to require transaction fees to be paid in a blockchain's native currency (much as U.S. taxes can only be paid using U.S. dollars). Specifically, Lecture 11 studies the design of transaction fee mechanisms, the component of a blockchain protocol that determines which transactions get executed and with what transaction fee. (Note that when demand for computation by a blockchain protocol exceeds its capacity, some transactions must get excluded.)

The lengthy Lecture 12 covers a completely different use case of a native cryptocurrency, namely to enable a Sybil-resistance mechanism different than (and in at least some respects superior to) the proof-of-work approach we studied in Lecture 9.

Finally, Lecture 13 focuses on the economic security of blockchain protocols—roughly, the cost of acquiring a majority or supermajority of the work/stake/etc. contributed to a protocol. We'll see there the importance to security of having a native currency with non-trivial real-world value.

(And of course, all of these reasons for a native currency are above and beyond Nakamoto's original one of creating a digital analog of physical cash.)

[2]Of course, whether such block rewards are economically meaningful depends on whether the going market price for the protocol's native currency is not too close to 0. Why that might or might not be the case is a longer discussion, outside the scope of this lecture.

[3]Block rewards are arguably most natural in longest-chain consensus protocols, because each block is produced unilaterally by a unique protocol participant. In a BFT-type protocol like Tendermint (Lecture 7), each block is the result of a collaboration between a large number of nodes (one node to propose a block and a supermajority to approve it). Accordingly, there's a range of options for rewarding nodes in such protocols, as will be discussed further in Lecture 12. (BFT-type protocols are typically coupled with proof-of-stake Sybil-resistance. Proof-of-stake protocols generally also need to incentivize node participation, because such participation typically requires devoting capital that could otherwise be used to earn interest.) In this lecture we'll narrow our focus to Nakamoto consensus protocols such as the Bitcoin protocol and the straightforward block rewards outlined above.

blocks—but do they really incentivize nodes to follow Nakamoto consensus as intended? That question will occupy us for the rest of this lecture (after a brief digression).

2.3 Digression: Block Rewards As Inflation

In the Bitcoin protocol, block rewards elegantly kill two birds with one stone. What motivates nodes to run the protocol? Block rewards. Where does money come from in the first place? Also block rewards. Thus block rewards solve both a microeconomic problem (specifying the game theory around mining) and a macroeconomic one (specifying the monetary policy for the protocol's native currency).

The Bitcoin protocol is remarkable, and unlike typical modern blockchain protocols, in that block rewards are literally the only way that Bitcoins (the currency) have even been created. When the protocol was first deployed, there were zero Bitcoins in existence. When the first block (following genesis) was mined (presumably by Nakamoto themself), 50 Bitcoins (the block reward at the time) were printed into existence. Roughly 10 minutes later, when the second block was mined (presumably again by Nakamoto), another 50 Bitcoins were born. And so on, up until this very moment. Block rewards (and hence the inflation rate of the Bitcoin cryptocurrency) are programmed to decrease over time, eventually reaching zero a little over a century from now, leaving the world with 21 million and only 21 million Bitcoins in existence.

This hard supply cap on the money supply seems to have been an important feature to Nakamoto, perhaps as a reaction (and intended correction) to the money-printing used by various governments for bank bailouts and other purposes during the Great Recession in 2008. Some modern blockchain protocols embrace Bitcoin's hard supply cap philosophy (with inflation eventually going to 0), while others mimic the monetary policy of fiat currencies (which typically allows for perpetual inflation to help drive growth).[4] There is currently little understanding as to which approach is "better," or as to how the features and goals of a specific blockchain protocol might inform the protocol's monetary policy.

Hard supply cap or not, modern blockchain protocols generally launch with an "initial distribution" of the native currency to the protocol's various stakeholders, which may then be further inflated over time (e.g., through block rewards). As you can imagine, much blood and ink gets spilled over the details of the initial distribution, but generally speaking some goes to the team that developed the protocol, some to investors, and some to the "community" (in the last case, perhaps via "airdrops" to those who participated in or used preliminary versions of the protocol).

2.4 Selfish Mining: The Perils of Misguided Incentives

Adding a native currency to a blockchain protocol solves some problems but also introduces some new ones. Whenever you introduce new incentives into a system, and whatever your

[4]The inflation rates currently used by different major blockchain protocols also differ significantly, but most lie in the 2-8% annual range.

intuition may be about what those incentives achieve, it's important to take a step back and ask: "Wait, what are people *actually* incentivized to do now that I've introduced this new incentive system?"

In Nakamoto consensus, block rewards are intended to motivate the nodes running the protocol to follow it as intended, meaning work to solve cryptopuzzles in order to extend the longest chain with a new block (and, upon producing such a block, broadcasting it to everyone else). But do block rewards motivate the nodes even more strongly to behave in some other, unintended way?

With economically meaningful rewards, the traditional dichotomy between "honest" and "Byzantine" nodes used in all previous lectures is no longer appropriate. There might be a few honest nodes (that follow the protocol no matter what the incentives are, e.g. if run by the founding team) and some Byzantine nodes (who only care about disrupting the protocol), but in a permissionless system you must expect many participants to act in their own self-interest and behave in whatever way maximizes their economic reward from the protocol. So with block rewards in Nakamoto consensus, the right question to ask is: is honestly following the protocol a profit-maximizing strategy for a node?

This lecture explains why the answer to this question is "no." Specifically, we'll see that, in a range of scenarios, a strategy known as "selfish mining" can garner more rewards for a node than obediently carrying out Nakamoto consensus as intended. While the attack is somewhat specific to both longest-chain consensus (using forking attacks reminiscent of Lecture 8) and proof-of-work difficulty adjustment (see Section **??**),[5] you should think of it as a famous case study of the catch-22 faced by blockchain protocol designers. Such protocols need incentives to encourage correct behavior by its participants, but even minor flaws in their design can strongly encourage unintended (and undesirable) behavior.

3 How Can a Node Maximize Its Block Rewards?

For the rest of this lecture, let's think specifically about Nakamoto consensus with a fixed block reward that is given to each producer of a block on the longest chain.[6] As a profit-maximizing node, what you should you do?

The subtle issue: non-uniform orphaning rates. Your first thought might be, does it really matter? After all, by the nature of proof-of-work sybil-resistance (under the random oracle assumption), a node with (say) 10% of the overall hashrate can generate only 10% (on average) of the overall blocks, which would seem to lead to a 10% share of the block rewards.

The subtle point is that a node garners rewards only for the blocks it produces that wind up on the longest chain—if it produces a block that gets orphaned, it gets no compensation

[5]This attack was analyzed back in 2013 when Bitcoin was more or less the only game in town, so the focus on Nakamoto consensus should come as no surprise.

[6]Or, more precisely, of a block that is sufficiently deep on the longest chain (as quantified by the security parameter k from Lectures 8 and 9).

for its efforts. If every node follows the protocol honestly, then the longest chain grows in an orderly fashion, without any forks (for simplicity, assume that message delays are small and so there are no inadvertent honest forks). Because every block ever created winds up on the longest chain, producing 10% of the blocks overall means producing 10% of the blocks on the longest chain and hence earning 10% of the overall block rewards. But could it be that deviations from honest behavior could lead to different nodes getting their blocks orphaned at different rates? This would be a problem, because nodes with higher-than-average orphan rates would earn a lower-than-expected share of the block rewards.

Difficulty adjustment review. To make the potential issue more concrete, let's page back in some details about how difficulty adjustment works in Nakamoto consensus. Recall from Lecture 9 that puzzle solutions are inputs x that—in addition to meeting various formatting constraints—hash to something close to 0: $h(x) \leq \tau$, where h is a cryptographic hash function like SHA-256 and τ is the difficulty threshold. The parameter τ is maintained by the protocol, and is typically tuned to target a particular block rate (e.g., one per ten minutes in the Bitcoin protocol).

Actually, the description above—the usual one that you'll hear—is slightly misleading. In typical Nakamoto consensus, nothing matters other than the blocks on the longest chain. In particular, blocks off of the longest chain are ignored when adjusting the difficulty threshold. For example, in the Bitcoin protocol, for every new batch of 2016 blocks on the longest chain, the protocol uses those blocks' timestamps to estimate the amount of time that elapsed during their production. If, according to their timestamps, these 2016 blocks took more than two weeks to produce, then τ is adjusted higher (making the cryptopuzzles easier). If the latest batch of 2016 blocks took less than two weeks to produce, than τ is decreased (to make the puzzles harder). (Why 2016? Because if the target is to have the longest chain grow by one block on average every ten minutes, then you're hoping to see $6 \times 24 \times 14 = 2016$ blocks in a two-week period.) The protocol does not differentiate between the case that these 2016 blocks were the only blocks created in that period (all winding up on the longest chain), and the case that 4032 blocks were created during that time (with half on the longest chain and half orphaned). The rate of overall block production is twice as fast in the second scenario, but the rate of growth in the longest chain is the same either way.

Block rewards as a fixed-size pie. Block rewards, remember, are also doled out only to the block on the longest chain (and not to any orphaned blocks). Thus by targeting a rate of growth of the longest chain, Nakamoto consensus difficulty adjustment also targets a rate of minting block rewards. In the case of the Bitcoin protocol, with the current block reward of 6.25 BTC, the target is 12600 BTC per two weeks.

Because Nakamoto difficulty adjustment effectively prevents a node from manipulating the size of the pie (e.g., 12600 BTC/fortnight), at least in the long run, a profit-maximizing node should act to maximize the size of its slice. This means that the node should act to maximize the *fraction* of the blocks on the longest chain that it produced, and therefore the fraction of the (fixed-size) pie of block rewards that it earns.

Goal of a Profit-Maximizing Node

Maximize its fraction of the blocks on the longest chain.

You might hope that this pie gets split proportionally among the nodes according to their hashrates. By the random oracle assumption, a node with 10% of the overall hashrate will produce (on average) 10% of the blocks. Does this also translate to producing (on average) 10% of the blocks on the longest chain? Or could a particularly devious 10% node somehow be responsible for 11% of the blocks on the longest chain?

Honesty is not the best policy! The key takeaway of this lecture, which is neither obvious nor intuitive, is that a node in Nakamoto consensus can in many cases boost its share of the block rewards by deviating from the behavior intended by the protocol. For example, if a node has 10% of the overall hashrate and superior network connectivity, it actually can guarantee itself 11% of the blocks on the longest chain (assuming that the other nodes all follow the protocol honestly).

The types of deviations discussed in this lecture sometimes go by the name *selfish mining.* The "selfish" part refers to the (realistic!) assumption that nodes are neither mindlessly obedient nor Byzantine, but rather act in their own interest. "Mining" refers to block production in proof-of-work blockchain protocols (with the loose analogy that trying lots of different nonces is like digging for gold).[7]

For the game theory fans out there, this lecture's key takeway can be summarized as:

> In Nakamoto consensus, obediently following the protocol does not generally constitute a Nash equilibrium.

A Nash equilibrium refers to an outcome of a game (i.e., a choice of strategy for each of the game's players) in which no one has an incentive to deviate unilaterally to a different strategy. An outcome (like all nodes behaving as intended in Nakamoto consensus) is not a Nash equilibrium, then, if one of the participants *is* better off (e.g., earning a larger share of block rewards) by unilaterally changing its strategy. And this is exactly what we'll show in this lecture.[8]

Given that the Bitcoin protocol has been happily humming along for fourteen-plus years, it would seem that selfish mining attacks are not exactly a fatal flaw; see Section **??** for a detailed discussion of their practical implications. Primarily, you should interpret the results in this lecture as a cautionary tale. Whenever you add incentives to a protocol, you may well have strong intuition about why they incentivize participants to do what you want. But

[7]Sometimes you'll see "selfish mining" refer more generally to deviations from intended behavior by a block producer in a blockchain protocol. Our focus here is on the original and narrower meaning of term (referring to a specific type of deviation in Nakamoto consensus, based on deliberate forks and delayed block announcements).

[8]Reading between the lines in the Bitcoin white paper [**?**], it seems plausible that Nakamoto assumed that honestly following the protocol would in fact be a Nash equilibrium (at least, under the assumption that no single node has 50% or more of the overall hashrate).

more often than not, especially when there is a rich set of deviations available, your intuition will be misguided and the incentives will encourage the most clever participants to behave in unintended and undesirable ways.

Where we're going. This lecture presents three versions of the takeaway above, each more complex and compelling than the last. In all three versions, we'll work in the "super-synchronous" or "instant communication" model from Lecture 8, meaning that every message will arrive at its destination immediately (as in the synchronous model with maximum message delay $\Delta = 0$). This is, of course, an unrealistic approximation a real communication network. Back in Lectures 8 and 9, we were proving positive results (consistency, liveness, etc.) about longest-chain consensus, and we adopted this assumption only temporarily, for expository convenience. (Toward the end of Lecture 9, we outlined how to extend those positive results to the general synchronous model.) In this lecture, by contrast, we're proving *negative* things about Nakamoto consensus and thus will work in the unrealistic super-synchronous model with pride. We'll prove that *even if* non-Byzantine nodes can magically communicate with each other instantly, nodes are generally incentivized to deviate from their intended behavior.

We'll start by warming up in Section 4 with the extreme case of a node that controls 51% of the overall hashrate. Here, it will be relatively simple to understand how such a node can benefit from deviating from honest behavior. While the node's deviating strategy in this case is relatively simple, it nevertheless carries the seeds of the more complex deviations used in the other two scenarios.

In the second and third scenarios, we consider only nodes that have less than 50% of the overall hashrate. The difference between the two scenarios concerns how honest nodes break ties between multiple chains that are equally long. (Remember, in longest-chain consensus, an honest node is allowed to extend an arbitrary longest chain.) First, in Section 5, we'll set things up to be as favorable as possible toward deviations away from honesty—when a node contemplates a deviation, we'll assume that it also gets to choose how the other (honest) nodes break ties throughout the course of the protocol. This is not necessarily a realistic scenario, but it is also hard to rule out—e.g., the deviation that we'll describe could be well approximated by a deviating node that has superior network connectivity under the assumption that honest nodes break ties according to the block that they heard about first (which is in fact the official tie-breaking rule of the Bitcoin protocol). In this scenario, we'll see that, no matter how little hashrate the deviating node has, if all the other nodes behave honestly, then the node is better off mining selfishly than honestly.

Finally, in Section 6, we'll consider the most realistic scenario, of a node with less than 50% of the overall hashrate that cannot control how the other nodes break ties among competing longest chains. This setup is maximally unfavorable to a deviating node, and indeed, nodes with sufficiently small hashrate are best off following Nakamoto consensus honestly. If a node is big enough (at least roughly 33% of the overall hashrate), however, it can overcome the losses from adversarial tie-breaking and improve over honest mining with a sophisticated selfish mining strategy.

Punchline

The incentive-compatibility threshold of Nakamoto consensus ($\approx 33\%$) is strictly smaller than the security threshold (50%).

Here by "security threshold" we mean the upper bound that must be imposed on the fraction of hashrate controlled by a non-honest node (or a cartel of such nodes) in order for Nakamoto consensus to be consistent and live. By "incentive-compatibility threshold," we mean the analogous upper bound in order for honest behavior to constitute a Nash equilibrium.

4 A Profitable Deviation with 51% of the Hashrate

The setup. Let's start with an extreme case in which it's relatively easy to see why nodes are not always incentivized to obediently follow Nakamoto consensus. In this section, we'll make the following assumptions:

1. All messages are delivered instantly (i.e., the super-synchronous model).
2. A deviating node can control how other nodes break ties among competing longest chains.
3. There is a node A that controls (at least) 51% of the hashrate.[9]
4. Every node other than A obediently follows Nakamoto consensus.

The first assumption only makes our (negative) results stronger, so we're happy to make it. However we may feel about the second assumption, in this section we're making it only for simplicity. In the presence of the third assumption, it's relatively easy to extend the arguments in this section to the case where the deviating node A cannot control how other nodes break ties (a good exercise for the reader). The third assumption is extreme, and we'll drop in both Section 5 (while retaining the second assumption) and Section 6 (while also dropping the second assumption). The fourth assumption corresponds to our goal of showing the honest behavior does not constitute a Nash equilibrium: *even if* all the other nodes behave honestly, that honesty is not necessarily contagious.

You'd be right to object that, as we saw in Lectures 8 and 9, Nakamoto consensus doesn't have any desirable properties (such as consistency and liveness in the synchronous setting) when a (possibly Byzantine) node controls at least 51% of the hashrate. Why should we care about this case? In some ways we don't, but as we'll see, we'll get very accurate intuition about all the arguments in this lecture by studying this extreme case.

[9]In previous lectures, we used A to denote an "adversary," which we thought of as a node that wanted to interfere with the protocol. Here, node A is a humble profit-maximizer trying to maximize its block rewards, not a Byzantine node per se.

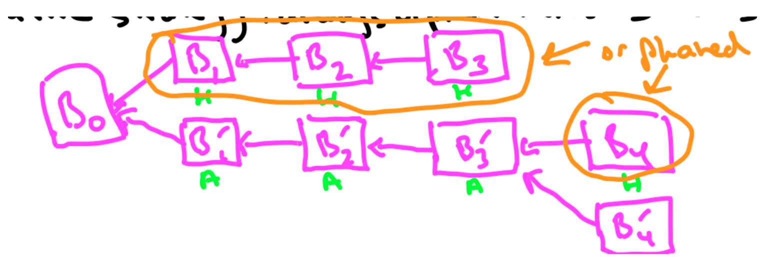

Figure 1: A node with 51% of the hashrate can orphan every block every produced by the other nodes.

Rewards earned by honest behavior. Node A always has the option of obediently following the protocol. If it does so, because all other nodes honestly follow the protocol also (by assumption) and all messages are delivered instantly (again, by assumption), the longest chain will grow in an orderly fashion, without any forks. That is, every block produced winds up on the longest chain. Because A has 51% of the hashrate, it produces 51% of the blocks (in expectation) and therefore its blocks make up 51% of the longest chain (in expectation).

For example, if we focus on the Bitcoin protocol, with 2016 BTC allocated in a typical two-week period, honest behavior will net node A $0.51 \times 2016 \approx 1028$ BTC over the same period (in expectation). Can node A do better?

Deviating to fully control the longest chain. A 51% node can, in fact, guarantee itself 100% of the block rewards! Thus, honesty is definitely not the best policy for such a node.

To control 100% of the blocks on the longest chain, node A must somehow manage to orphan every block ever produced by any other node. How can it do this? Let's see with an example (Figure 1). Suppose we fire up Nakamoto consensus with genesis block B_0, and all nodes (both node A and all the honest nodes) start dutifully attempting to solve the protocol's cryptopuzzles. Node A, by virtue of contributing 51% of the overall hashrate, has a 51% chance of producing the next block. But for this example, let's suppose some honest node gets lucky (which happens with 49% probability) and produces a block B_1 that extends the genesis block B_0.

Block B_1 is now the end of the longest chain, so the honest nodes will stop trying to extend B_0 and will start trying to extend B_1 instead. Node A, however, is intent on forcing block B_1 out of the longest chain. (Remember, node A is shooting for 100% of the block rewards, which entails preventing any other node from getting a block on the longest chain.) The only way node A can orphan B_1 is to create an alternative longest chain that excludes B_1. Thus, node A cannot work to extend B_1, and must instead continue trying to extend the genesis block B_0.

Next, there's again a 51% chance that node A is the next one to produce a block (extending B_0) and a 49% chance that some other node is the next one to produce a block (extending B_1). For this example, let's suppose that the latter happens again, with some honest node producing a block B_2 that extends B_1. Honest nodes then switch to working to extend the new end of the longest chain (i.e., B_2).

Node A can work to extend any of B_0, B_1, or B_2. But again, if its ambition is to eventually orphan the honestly produced block B_1, it has no choice but to continue trying to grow an alternative chain emanating out of B_0.

Next, there's again a 51% chance that the next block is produced by node A (extending B_0) and a 49% chance that some other node produces it (extending B_2). Suppose node A finally succeeds in producing a block B'_1 that extends the genesis block B_0. Node A's work is certainly not done; at this point, blocks B_1 and B_2 still reside on the unique longest chain. Honest nodes continue to try to extend the tip B_2 or the longest chain, while node A naturally switches to trying to extend its alternative chain further (working to extend B'_1).

Suppose the next block is produced by an honest node (a 49% change)—a block B_3 that extends B_2. Honest nodes switch to trying to extend B_3. Undeterred, node A continues in its quest to orphan all the blocks produced by honest nodes, trying to extend its block B'_1.

Let's suppose that what happens next is that node A successfully produces a block B'_2 extending B'_1 before any honest node produces a block (a 51% chance) and then gets lucky again (another 51% chance), producing a block B'_3 that extends B'_2. Now, there is a tie for the longest chain and, under second assumption above (about tie-breaking), blocks B_1, B_2, and B_3 are as good as orphaned. Why? Node A, obviously, will next try to extend B'_3, the tip of its chain. Honest nodes might extend either block B_3 or B'_3. But under assumption that node A can choose how honest nodes break ties, it can choose for all honest nodes to also try to extend B'_3. With all nodes trying to extend B'_3 and ignoring B_3, blocks B_1, B_2, and B_3 have no chance of ever winding up on the longest chain.

The next block will be one that extends B'_3. Let's suppose (a 49% chance) that a honest node produces the next block, a block B_4 that extends B'_3. Node A now has to go back into orphaning mode. To ensure that B_4 does not wind up on the longest chain, node A must create an alternative chain that orphans it. The obvious strategy for node A is to continue trying to extend B'_3, even as honest nodes switch to trying to extend B_4. If it succeeds (a 51% chance) in creating an alternative block B'_4 that extends $b"_3$, then B_4 is guaranteed to be orphaned (node A will never extend it and, because node A also controls honest nodes' tie-breaking, no honest node will ever extend it either).

Hopefully the general strategy for node A is now clear. Whenever an honest node produces a block B (necessarily at the tip of the longest chain), node A stubbornly attempts to create an alternative longest chain, beginning from node A's most recently produced block. Node A does not stop until succeeds. And succeed it must—with 51% of the hashrate, it produces blocks more frequently (on average) than all the honest nodes combined. Node A's alternative chain will therefore eventually catch up to the honest chain, at which point all blocks produced by honest nodes will have been orphaned.[10]

Discussion. A node A with 51% of the overall hashrate can dutifully follow Nakamoto consensus results and produce 51% of the blocks on the longest chain, or mine selfishly and

[10]This is the role of the 51% assumption in this section. If node A produced blocks less frequently than honest nodes, there's a significant chance that its alternative chain would never catch up. In the next section, when we consider a node A with less than 50% of the overall hashrate, we'll obviously need to modify the selfish mining strategy.

produce 100% of the blocks on the longest chain. Recall from Section 3 that node A can't influence the rate of the growth of the longest chain (which stays effectively constant due to Nakamoto consensus difficulty adjustment) and therefore can't influence the rate of block reward issuance (which is directly tied to longest chain growth). In other words, because node A can't effect the size of the pie (e.g., 12600 BTC/fortnight), its goal is to maximize the size of its slice. And by selfish mining, node A can get all of the pie, not merely half of it. The overall rate of block production will be twice as fast when node A mines selfishly than when it mines honestly (e.g., roughly 4032 blocks per fortnight rather than roughly 2016, with the difficult threshold twice as high), but the growth rate of the longest chain will be the same in both scenarios (with only A's blocks on the longest chain when it mines selfishly, and all the blocks on the longest chain when node A mines honestly).

The argument above probably seems to rely heavily on the assumption that the deviating node A can control how honest nodes break ties among competing longest chains. But if you think about it, it's easy to modify the selfish mining strategy to work without this assumption. (Basically, node A just has to grow its alternative chain for one extra block. And because it has 51% of the hashrate, it will eventually be able to do this.) The analysis in the next section will, however, depend crucially on this assumption (which will then be removed in Section 6).

Nakamoto consensus doesn't guarantee consistency or liveness when there's a node that controls 51% of the overall hashrate. So it's much more interesting to consider selfish mining in the case where Nakamoto consensus *does* guarantee consistency and liveness, and that's the next order of business. The fact that selfish mining can boost a 51% node's share of block rewards from 51% to 100% makes you wonder if it might also be able to boost (say) a 10% node's share to some number bigger than 10% (even if well less than 100%), perhaps by selectively orphaning blocks produced by honest nodes. The rest of this lecture investigates to what extent this is the case.

5 Selfish Mining with Control over Honest Tie-Breaking

5.1 The Setup

In this section, we'll make the following assumptions:

1. All messages are delivered instantly (i.e., the super-synchronous model).
2. A deviating node A can control how other nodes break ties among competing longest chains.
3. Every node other than A obediently follows Nakamoto consensus.

This is the same set of assumptions as in the last section, except that we are now allowing the deviating node A to have an arbitrarily small fraction α of the overall hashrate. The first assumption only makes our (negative) results stronger. The second assumption is a strong one and crucial for this section's analysis (unlike in the previous section, where we adopted it only for convenience); the point of Section 6 to remove it. The third assumption reflects our goal of showing that node A may not be incentivized to follow Nakamoto consensus, even if all the other nodes choose to do so.

The main result of this section is:

Theorem 5.1 ([?]) *Under the three assumptions above, a node A that controls an $\alpha < \frac{1}{2}$ fraction of the overall hashrate can mine selfishly to guarantee itself a*

$$\frac{\alpha}{1-\alpha}$$

fraction of the overall block rewards.

For example, a node with 33% of the hashrate can guarantee itself 50% of the block rewards. A node with 10% of the hashrate can guarantee itself 11.1% of the block rewards. No matter how small α is, the deviating node is better off mining selfishly than honestly. (Recall that if node A mines honestly, it captures an $\alpha < \alpha/(1-\alpha)$ fraction of the block rewards.)

Theorem 5.1, and Theorem 6.1 in the next section, are academically pretty famous results. Last I checked, the original selfish mining paper of Eyal and Sirer [?] has more citations than any other formally published research paper about blockchain protocols. And on a personal note, this paper made a big impression on me at the time. I had always thought the Bitcoin protocol was strikingly elegant, but it was with this paper that I began to appreciate the depth of the computer science and game theory involved.

5.2 The Strategy

The plan. The goal of a profit-maximizing node A is to maximize its slice of the pie—the fraction of the blocks on the longest chain (see Section 3). Thus all else equal, it wants to do two things:

(i) Of the blocks that node A produces ("A-blocks"), get as many as them as possible into the longest chain.

(ii) Of the blocks that honest nodes produce ("H"-blocks), force as many as them as possible out of the longest chain.

In the last section, we saw how a node A with 51% of the overall hashrate could succeed perfectly at both goals simultaneously, with all of its blocks and none of the honest nodes' blocks included in the longest chain. Intuitively, the second property seemed to rely heavily on the 51% assumption, so that node A could be sure that it would eventually grow a long enough alternative chain to take over as the longest chain. Accordingly, the plan for this section's strategy is to continue to achieve the best-possible version of property (i) and, subject to this, to do as well as one can on property (ii). That is, we'll pursue a strategy that's guaranteed to get every A-block in the longest chain, and also at least occasionally manages to orphan an H-block. If we succeed, this strategy will automatically outperform honest mining. (The strategy gets as many A-blocks in the longest chain as honest mining

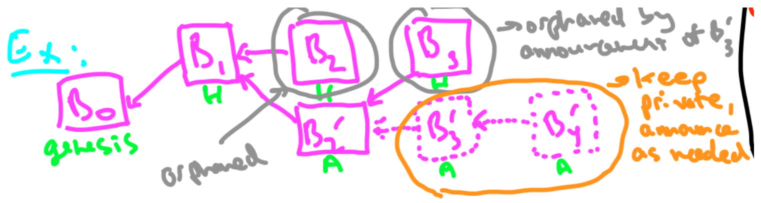
Ex:
B_0
genesis
B_1
H
B_2
H
B_3
H
orphaned by announcement of B_3'
B_2'
A
orphaned
B_3'
A
B_4'
A
keep private, announce as needed

Figure 2: A strategy that guarantees the inclusion of every A-block on the longest chain while also orphaning a positive fraction of the H-blocks.

(all of them) while orphaning more H-blocks, implying that the share of A-blocks in the longest chain with selfish mining must be higher that with honest mining.)

Speedy abandonment. Like in the previous section, let's introduce the strategy through an example (Figure 2). Start with a genesis block, B_0. There's an α chance the next block is an A-block, and a $1 - \alpha$ chance that it's an H-block. (Remember, α denotes the fraction of hashrate controlled by node A.) Let's suppose that the latter (the more likely event) happens, with some honest node producing a block B_1 that extends B_0. Honest nodes dutifully switch to trying to extend B_1, the tip of the longest chain. What should node A do? To have any chance of orphaning the H-block B_1, node A must attempt to grow an alternative chain starting from the genesis block B_0.

Next, there's an α chance that the next block produced will be an A-block (extending B_0) and a $1 - \alpha$ chance that it's an H-block (extending B_1). If node A succeeds in creating its A-block, then, because of our standing tie-breaking assumption, B_1 is as good as orphaned. (Node A will never extend B_1, and will break ties for the honest nodes so that they also never extend B_1.) But let's suppose that the second block is also an H-block, with B_2 extending B_1.

When this same sequence of events occurred in the last section, node A's strategy was to carry on and continue its attempt to create an alternative chain starting at B_0. There wasn't any real risk in doing so, because (by that section's assumption) node A produced blocks more frequently than all the honest nodes combined, and so it would eventually succeed in creating a new longest chain. Here, when node A has only an $\alpha < \frac{1}{2}$ fraction of the hashrate, it might never catch up (resulting in 0% of the block rewards!). Presumably, node A should give up at some point and abandon whatever alternative chain it's trying to grow, once its chances of catching up to the longest chain become sufficiently remote.

Given our plan of never allowing any A-blocks to get orphaned, node A needs to give up and reset quickly. (If it grew a non-empty alternative chain of A-blocks and ever stopped extending it, all of those A-blocks would get orphaned.) In our running example, with the H-block B_2 extending the H-block B_1 and no A-blocks yet created, node A concedes the H-block B_1 to the longest chain and channels its energies into instead orphaning B_2. (Such concessions are the price paid for owning less than 50% of the overall hashrate.) This entails trying to create a block B_2' extending B_1, which (with our standing tie-breaking assumption) would effectively orphan the H-block B_2. Let's suppose this is indeed what happens next, resulting in all nodes (node A and the honest nodes) trying to extend the block B_2'.

Delayed block announcements. So far, node A has only engaged in the usual longest-chain shenanigans of deliberate forking. What else could it do? We saw the answer back in Lecture 8, when we were analyzing the consistency and liveness properties of permissioned longest-chain consensus. In our proofs, we allowed a Byzantine node to choose whatever predecessors it wanted (possibly deviating from the longest chain) and also to keep secret any blocks that it has created, perhaps saving them for release at a later time. At the time, it wasn't clear why a Byzantine node would want to delay a block announcement (though it certainly could if it wanted), and in fact the threshold of honest hashrate necessary for consistency and liveness (51%) was the same whether Byzantine nodes delayed block announcements or not. In that lecture, this additional power only served to make our proofs trickier. Here, we'll see how delayed block announcements are directly useful to a node striving to maximize its share of the block rewards.

Resuming our running example where we left off, suppose node A just succeeded in creating the A-block B_2', thereby orphaning the H-block B_2. Every node is now trying to extend the block B_2', and there's an α chance that node A is the first to do so. Suppose this does in fact happen, with A-block B_3' extending B_2'. Node A could announce B_3' immediately, and this would guarantee that block's inclusion in the longest chain. But there's something more strategically clever that it could do, which is to keep quiet about its new block B_3'. Why is this helpful? *Because it tricks the honest nodes into trying to extend a block that is not the tip of the longest chain*, namely B_2' (the tip of the longest chain that honest nodes happen to know about); meanwhile, node A can secretly work to extend B_3' with yet another A-block.

Imagine that the next block created is an H-block. Had node A announced its new block B_3' to everyone, this H-block would have been a block B_4 extending B_3'. This block would be the new tip of the longest chain, and would have a legitimate shot at staying in the longest chain forevermore. But because node A kept the block B_3' secret, the new H-block is instead a block B_3 extending the old block B_2'. If node A plays its cards right, there's no chance that this H-block winds up in the longest chain: node A can announce its secret block B_3' immediately after B_3 is announced, at which point (because of our standing tie-breaking assumption) B_3 is as good as orphaned. In effect, keeping B_3' in its back pocket allowed node A to nullify block B_3 as soon as it was announced.

In general, node A will make sure that each block that it creates is put to good use, orphaning some H-block at the same block height. (Remember, the height of a block is the number of hops between it and the genesis block. In our example, we always number blocks according to their height.) In our running example, suppose that the block created after the (currently secret) A-block B_3' is not an H-block but another A-block, a block B_4' that extends B_3'. Node A keeps B_4' (along with B_3') to itself, and works to extend B_4' with yet another A-block (while the poor honest nodes remain in the dark, trying to extend B_2'). If at some point an honest node finds a block B_3 extending B_2', then node A can orphan it immediately by announcement B_3'. (Node A keeps B_4' in its back pocket.) If the honest nodes get lucky again and produce the next block B_4 (this time, extending B_3'), then node A can immediately announce its block B_4' to nullify that block, as well.

The general strategy. The general strategy is exactly what you would think it would be, given our running example. Again, there are two main ideas: if the tip of the longest chain is an H-block, try to orphan it (conceding any H-blocks that are already buried deeper than the tip of the longest chain); and otherwise, keep A-blocks secret, releasing them on an as-needed basis to nullify subsequently created H-blocks. These two ideas show up in Case 2 and Case 1 below, respectively.

The Selfish Mining Strategy (with Control over Tie-Breaking)

Notation: Let h denote the maximum height of any block produced thus far (by either node A or by an honest node).

Case 1: If there is a height-h A-block, work to extend it.
[If successful, keep private.][11]

Case 2: If the only height-h block is an H-block B, then work to orphan it by extending B's predecessor with a competing height-h A-block.
[If successful, announce immediately.]

Throughout: Announce an A-block at height h (if one exists) as soon as a height-h H-block is announced.

In our running example, we saw Case 1 in action at the point when there was a height-2 A-block B_2' and no block at a larger height (and then again once the A-block B_3', extending B_2', was produced). We saw Case 2 in action at the point when there was a height-2 H-block B_2 and no height-2 blocks.

5.3 The Analysis

If every node obediently follows Nakamoto consensus, a node with an α fraction of the overall hashrate with produce (in the long run) an α fraction of the blocks on the longest chain and hence an α fraction of the overall block rewards. (In the super-synchronous model, every block produced will make it into the longest chain.) Can the selfish mining strategy in Section 5.2 boost a node's share of the overall block rewards? (We already discussed why the answer is clearly yes, given that the strategy guarantees that all A-blocks make it into the longest chain and that a nonzero number of H-blocks will get orphaned.) By how much?

For the analysis, think about a long sequence of N consecutive blocks produced by the nodes. (Think of N in the thousands, say. Some of these blocks will wind up on the longest chain, others may be orphaned.) Assume that a node with an α fraction of the hashrate produces exactly an α fraction of these N blocks. (Under the random oracle assumption, as N grows large, the law of large numbers tells us that this will typically be the case, up to negligible deviations. See also our "proportional representation" arguments from Lecture 8.) The analysis then has three steps.

[11]This case applies also at the beginning of the protocol, when the only block is the genesis block.

Step 1. The first observation is simply that αN of the N blocks above will be A-blocks, with the remaining $(1-\alpha)N$ blocks being H-blocks. The question, then, is how many of each wind up on the longest chain.

Step 2. The next claim is that, as telegraphed earlier, 100% of the A-blocks produced wind up on the longest chain—a perfect implementation of property (i) in the original plan in Section 5.2. Why is this true? Well first of all, by construction of the strategy, whenever an A-block is announced, that block is at that time the tip of one of the longest chains known to honest nodes. In our running example, the A-block B_2' is announced as soon as its created, and is tied with the H-block B_2 for the tip of the longest chain. The A-blocks B_3' and B_4' are announced only (immediately) after the announcement of the H-blocks B_3 and B_4 at the same heights, at which times they are again tied for the tip of the longest chain known to honest nodes. In general, an A-block is announced either in Case 2 of the strategy (where by construction the block is tied with the incumbent tip of the longest chain) or on an as-needed basis to nullify a just-announced H-block at the same height (as in the "Throughout" part of the strategy). Also, every A-block will be announced eventually (once honest nodes eventually manage to produce an H-block at the same height).

Given that every A-block is, upon announcement, the tip of a longest chain, and given our standing tie-breaking assumption, every A-block is guaranteed inclusion in the longest chain. When a new A-block B' is announced, certainly node A will work to extend it; and because B' is the tip of a longest chain, node A can also force the honest nodes to work to extend it. Thus, every node will forevermore work to extend B' and its descendants, and B''s position is the longest chain is assured.

Step 3. What about property (ii) from the original plan in Section 5.2? How many H-blocks make it into the longest chain? Note that not all of them will be orphaned—in our running example, the H-block B_1 made it into the longest chain.

The key claim is that every A-block is responsible for orphaning exactly one H-block (at the same height). In our running example, the A-blocks B_2', B_3', and B_4' orphan the H-blocks B_2, B_3, and B_4, respectively. (In the case of B_2' the orphaning happens immediately; for B_3' and B_4' it happens eventually, once honest nodes get around to producing B_3 and B_4.) In general, as we saw in Step 2, whenever an A-block is announced, it is at that time tied (with some H-block) for the tip of the longest chain known to honest nodes. We also saw in Step 2 that, due to the current tie-breaking assumption, such an A-block is guaranteed to wind up on the longest chain. In turn, this means that the competing H-block is guaranteed to *not* wind up on the longest chain. (A chain can have only one block at each height.) Thus, for every A-block (on the longest chain), we can point to an H-block at the same height that gets excluded from the longest chain. Because all the A-blocks produced have distinct heights (an easily observed property of the selfish mining strategy), all of these orphaned H-blocks must be distinct.

Final calculations. Here are the takeaways from steps 1–3:

1. N blocks produced overall, with αN being A-blocks and $(1-\alpha)N$ being H-blocks.

2. All αN A-blocks wind up on the longest chain.

3. Each of the αN A-blocks forces a distinct H-block off of the longest chain.

With $(1-\alpha)N$ H-blocks total and αN of them knocked off the longest chain by a competing A-block, $(1-\alpha)N - \alpha N = (1-2\alpha)N$ H-blocks make it into the longest chain. Thus, the longest chain will have αN A-blocks, $(1-2\alpha)N$ H-blocks, for a total of $\alpha N + (1-2\alpha)N = (1-\alpha)N$ blocks on the longest chain. The fraction of A-blocks on the longest chain—which, by the discussion in Section 3, is what we actually care about—is

$$\frac{\text{\# of A-blocks on longest chain}}{\text{total \# of blocks on longest chain}} = \frac{\alpha N}{(1-\alpha)N} = \frac{\alpha}{1-\alpha}.$$

This completes the proof of Theorem 5.1.

Again, the key point is that $\alpha/(1-\alpha) > \alpha$ for every $\alpha > 0$. That is, no matter how little hashrate node A has, it is better off selfish mining than honest mining (assuming it can control how honest nodes break ties between competing longest chains, and assuming that all other nodes obediently follow Nakamoto consensus). A node with 20% of the hashrate can secure 25% of block rewards, a node with 10% of the hashrate 11.1% of the block rewards, and so on.

Discussion. You might be bothered by how strongly this section's analysis depends on the assumption that the deviating node can control how the honest nodes break ties. One thing to remember is that scenario is hard to entirely rule out—control over tie-breaking can perhaps be well approximated by superior network connectivity (e.g., if honest nodes break ties according to which block they heard about first).

That said, the point of Section 6 is to re-analyze selfish mining without this assumption. We'll see there that selfish mining can still improve over honest mining, but only for nodes that control a sufficiently large fraction of the overall hashrate (roughly 33%).

5.4 Connection to Chain Quality

Theorem 5.1 is not the first time that we've seen the expression $\alpha/(1-\alpha)$. Back in Lecture 8, we proved that (under a list of assumptions) Nakamoto consensus satisfies (probabilistic) consistency and liveness assuming that at least 51% of the overall hashrate is controlled by honest nodes. We then proceeded to ask if we could strengthen the liveness guarantee under stronger assumptions about the fraction of hashrate controlled by honest nodes. This led to the concept of *chain quality*, defined as the fraction of blocks on the longest chain that were contributed by honest nodes. (Liveness is basically equivalent to the guarantee that the chain quality is strictly positive, because Byzantine nodes can produce empty blocks while honest nodes are supposed to include in a block all the pending transactions that they know about.) We saw in that lecture that the proof of liveness for Nakamoto consensus was easily extended to show the following (with high probability over long block sequences, and ignoring a small amount of variance): if an $\alpha < \frac{1}{2}$ fraction of the hashrate is controlled by Byzantine nodes, then no matter how honest nodes break ties among competing longest chains, the chain quality is at least

$$\frac{1-2\alpha}{1-\alpha}.$$

At the time, we were disappointed that the chain quality guarantee was $(1-2\alpha)/(1-\alpha)$ rather than the (bigger) number $1-\alpha$. If 67% of the hashrate is controlled by honest nodes, why shouldn't the chain equality also by 67%?

Theorem 5.1 explains why. Suppose there's a Byzantine node that controls an α fraction of the overall hashrate, with the other $1-\alpha$ fraction controlled by honest nodes. One strategy available to this Byzantine node is to play the role of a profit-maximizing node A and carry out the selfish mining strategy in Section 5.2. Theorem 5.1 then implies that if ties between A-blocks and H-blocks at the same height are always broken in favor of A-blocks, this Byzantine node can guarantee itself an $\alpha/(1-\alpha)$ fraction of the blocks on the longest chain. The honest nodes are stuck with the leftovers, which is a

$$1-\frac{\alpha}{1-\alpha}=\frac{1-2\alpha}{1-\alpha}$$

fraction of the blocks on the longest chain.

Thus, the chain quality guarantee of $(1-2\alpha)/(1-\alpha)$ for Nakamoto consensus from Lecture 8 is not an artifact of our particular proof; there really is a strategy for Byzantine nodes to force the chain quality down to that level. Similarly, the guarantee of $\alpha/(1-\alpha)$ in Theorem 5.1 is not an artifact of its proof or of the specific selfish mining strategy described in Section **??**; a profit-maximizing node will be unable to drive the corresponding chain quality any lower. Each of these results—one about the game theory and the other about the security of Nakamoto consensus—shows that the other is the strongest possible (for every value of $\alpha \in (0, \frac{1}{2})$) and cannot be improved with more clever arguments or strategies (at least, without further assumptions about how honest nodes break ties).[12]

6 Selfish Mining: The Main Result

This section is easily the most mathematically difficult of this lecture, so you skip it if you're not excited about some interesting mathematical analysis. (But don't skip the discussion in Section **??**!) If you *do* like non-trivial mathematical analysis, like I do, this should be your favorite part of the lecture. The result in this section is what is usually considered the "main result" of the paper by Eyal and Sirer [**?**] mentioned earlier. The prevailing intuition prior to that paper was that honest mining should be optimal for a node as long as it has less than 50% of the overall hashrate, so this result came as a big surprise at the time.

[12]Note that comparing the two results is an apples-to-apples comparison, in the sense that both were proved in the super-synchronous model with honest nodes breaking ties in the way least favorable to them.

6.1 The Setup

For our final setting, we'll make the following assumptions:

1. All messages are delivered instantly (i.e., the super-synchronous model).

2. Honest nodes break ties among competing longest chains arbitrarily.

3. Every node other than A obediently follows Nakamoto consensus.

The difference with Section 5 is that ties between longest chains are no longer broken by node A, but arbitrarily. Concretely, this means that in the analysis in this section, we'll assume that ties between an H-block and an A-block at the same height will always be broken in favor of the H-block. Remember, the first assumption only makes our (negative) results stronger, and the third assumption reflects our goal of showing that honest mining is not a Nash equilibrium (i.e., honest mining may not be properly incentivized, even if all other nodes are mining honestly).

The main result of this section is:

Theorem 6.1 ([?]) *Under the three assumptions above, a node A that controls an $\alpha > \frac{1}{3}$ fraction of the overall hashrate can mine selfishly to guarantee itself strictly more than an α fraction of the overall block rewards.*

We'll be able to precisely quantify the gory details of "strictly more" in the analysis (Section **??**); the statement of Theorem 6.1 emphasized the main point, that selfish mining is superior to honest mining for nodes with a sufficiently large (but still less than 50%) fraction of the overall hashrate, no matter how honest nodes break ties between competing longest chains. (Recall that if node A mines honestly, it captures an $\alpha < \alpha/(1-\alpha)$ fraction of the block rewards.)

6.2 The Risks of Selfish Mining

The two goals. Remember that the goal of a profit-maximizing node A is to maximize its slice of the pie—the fraction of the blocks on the longest chain (see Section 3). Thus all else equal, it wants to do two things:

(i) Of the blocks that node A produces ("A-blocks"), get as many as them as possible into the longest chain.

(ii) Of the blocks that honest nodes produce ("H"-blocks), force as many as them as possible out of the longest chain.

We saw in Section 4 that a node with 51% of the hashrate has a strategy that perfectly implements (i) and (ii), with the longest chain consisting of all and only the A-blocks (i.e., with all H-blocks orphaned and no A-blocks orphaned). We saw in Section 5 that a node with an α fraction of the hashrate and control over the tie-breaking of honest nodes can again implement (i) perfectly (with all A-blocks included in the longest chain) and (ii) approximately (with the fraction of orphaned H-blocks increasing with α).

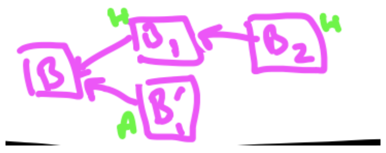

Figure 3: Any attempt to orphan an H-block runs the risk of a later orphaning of an A-block.

Example: the conflict between the two goals. In this section, with a node A with an $\alpha < \frac{1}{2}$ fraction of the hashrate and no control over tie-breaking, *properties (i) and (ii) are fundamentally in conflict.* That is, any attempt by node A to orphan any H-blocks will automatically introduce a risk that one or more A-blocks will themselves get orphaned—progress on goal (ii) requires compromises on goal (i).

o For example, consider Figure 3, and suppose first that only the blocks B, B_1, and B_1' are in existence. Suppose B_1 is an H-block and B_1' is an A-block. (Perhaps B_1 was produced first and node A then worked to produce B_1' with the hopes of orphaning B_1.) Last section, node A would have declared victory at this point—it's only going to extend its own block B_1' and, because it could dictate how honest nodes break ties, node A could also force all honest nodes to work to extend B_1' rather than B_1. In that setting, B_1 was as good as orphaned and B_1''s place in the longest chain was assured.

In this section, however, node A cannot force honest nodes to extend B_1' rather than B_1, and in fact must be prepared for the scenario in which all the honest nodes are doing the opposite. Which means that node A should be a bit nervous, right? It's trying to extend its own block B_1'; it can rest easy if it succeeds and produces another A-block B_2', as B_1' and B_2' would then belong to the unique longest chain, their places in the eventual longest chain secured (and B_1 orphaned). (If there's a unique longest chain, honest nodes must work to extend it.)

But, if honest nodes produce a block B_2 (extending B_1) first, as in Figure 3, then node A finds itself in a deeper hole than before. Now it needs to produce two blocks—a B_2' and a B_3'—in order to create a unique longest chain with its A-blocks. Meanwhile, the honest nodes will continue to dutifully try to extend the longest chain, and might well create a block B_3 (extending B_2) before node A is able to create B_2'. Indeed, because $\alpha < \frac{1}{2}$, in the long term, the combined block production of the honest nodes will outpace that of node A. Presumably, once node A finds itself in a deep enough hole—imagine it's 100 blocks behind the longest chain, say, with the chance of ever catching up astronomically small—it will give up and reset (e.g., switch to trying to orphan the tip of the current longest chain, rather than the whole thing). And whenever it does give up, it abandons all of the A-blocks it produced on its alternative chain (like the block B_1' in Figure 3). This is the sense in which, without control over tie-breaking, any attempt by a deviating node to orphan an H-block automatically carries the risk of orphaned A-blocks.

Implications. A node contemplating a deviation from Nakamoto consensus must weigh its benefits (in the form of orphaned H-blocks) against its costs (orphaned A-blocks), a tug-of-war between two complex forces. In Section 5 it was obvious that the selfish mining strategy in Section 5.2 would boost block rewards (the number of A-blocks on the longest chain stayed the same, the number of H-blocks was strictly smaller). Here, without control over tie-breaking, it's not obvious at all whether any selfish mining strategy could be profitable. (And, for nodes with a sufficiently small fraction of the hashrate, they're not.) And if selfish mining *can* improve over honest mining for sufficiently large hashrates, we have to expect that it will be a complex analysis that shows it, an analysis that quantifies and compares the costs and benefits of selfish mining and identifies the crossover point of hashrate at which the latter exceeds the former. (And indeed, the selfish mining strategy in Section 6.3 is more complex than that in Section 5.2, and the analysis in Section **??** is much more sophisticated than that in Section 5.3.)

6.3 The Strategy

Description of strategy. The selfish mining strategy in Section 5.2 had two cases, depending on whether there was an A-block with height at least as large as every H-block. Here we'll need four, which roughly parameterize the extent to which node A's private chain of A-blocks is farther ahead than the longest chain known to honest nodes. For concreteness and reference, we'll state here the full strategy; the rest of the section talks through it and illustrates it via examples. In all four cases, "success" means that node A produces a block before any honest node, and "failure" means the opposite.

The Selfish Mining Strategy (with No Control over Tie-Breaking)

Notation: Let h_p and h_s denote the maximum heights of a block known to all nodes and of a block known only to node A, respectively.[13]

Case 1: $h_p > h_s$.
(I.e., the tip of the longest chains is public.)
Node A works to extend the longest chain.
[If successful, keep the new A-block private and proceed to Case 2.]
[If failure, switch to the new tip of the longest chain, stay in Case 1.]

Case 2: $h_s = h_p + 1$.
(I.e., node A is one block ahead of all publicly known longest chains.)
Node A works to extend its private A-block.[14]
[If successful, keep the new A-block private and proceed to Case 4.]
[If failure, proceed to Case 3.]

Case 3: $h_s = h_p$.
(I.e., there's a secret A-block that, were it announced, would be tied for the longest chain.)
Node A works to extend its private A-block.[15]
[If successful, announce the private A-block and the newly created block.]

[If failure, give up and return to Case 1.]

Case 4: $h_s \geq h_p + 2$.
(I.e., node A is at least two blocks ahead of all publicly known longest chains.)
Node A works to extend the tip of its private chain of A-blocks.
[If successful, keep the new A-block private and continue in Case 4.]
[If failure but still have $h_s \geq h_p + 2$, continue in Case 4; otherwise (if h_s is now $h_p + 1$) announce the entire private chain of A-blocks.]

The parameter h_p tracks the (height of the) tip of a longest chain known to honest nodes (p for "public"); honest nodes will, by definition, work to extend such a block. (Remember, the length of a chain equals the height of its tip.) Honest nodes know about all H-blocks ever produced (honest nodes immediately broadcast newly created blocks), and whichever A-blocks node A has deigned to announce. There may be additional blocks that node A has created but not announced; the parameter h_s tracks the maximum height of any of these blocks (s for "secret").

Next we'll talk through the four cases of the strategy. There are four relevant pieces of information for each: (i) the conditions under which the case applies; (ii) which block node A attempts to extend; (iii) what happens next if node A produces a block before any honest node does; (iv) what happens next if an honest node produces a block before node A does.

References

[13]If every block known to node A is also known to all honest nodes, interpret h_s as -1.

[14]As we'll see, the only way that this case can be reached is by node A successfully producing a (private) block in Case 1.

[15]As we'll see, the only way that this case can be reached is by node A successfully producing a (private) block in Case 1 and then an honest node successfully producing an H-block in Case 2.