CS269I: Incentives in Computer Science
Lecture #9: Incentives in Bitcoin Mining*

Tim Roughgarden†

October 24, 2016

1 Bitcoin

Digital currencies have been around for decades, but only over the past 7 or so years has one really taken off: *Bitcoin*. Bitcoin enables digital payments between untrusted parties in a fully decentralized way, meaning with no central authority involved (no banks, credit card companies, governments, etc.).[1] Before Bitcoin, it was not at all obvious that such a system could function at a reasonably large scale.

Why use Bitcoin? The jury is still out on what its killer applications will be. One prosaic but representative example is transferring money internationally. Ever tried to do that through a bank? You'll typically be charged a lot (like 20 USD) and the funds are usually released only after a couple of days. Through Bitcoin, you can accomplish the same goal, saving an order of magnitude in both time and cost.

So how does Bitcoin work? The basic primitive is that of a *transaction*, which includes the following ingredients.

A Bitcoin Transaction

1. One or more senders.
2. One or more receivers.
3. The amount of BTC (Bitcoins) transferred from each sender to each receiver.
4. A proof of ownership of the coins being transferred, in the form of a pointer back to most recent transactions involving the transferred coins.

*©2016, Tim Roughgarden.

†Department of Computer Science, Stanford University, 474 Gates Building, 353 Serra Mall, Stanford, CA 94305. Email: tim@cs.stanford.edu.

[1]Without the decentralization constraint, a system like Paypal already solves the problem.

5. A transaction fee, paid by the sender to the authorizer of the transaction.

Several comments. First, feel free to think only about transactions with a single sender and receiver (this is the common case, anyways). Second, senders and receivers are specified by their public keys. So the notion of a "user" of Bitcoin is equated with a specific public key.[2] Third, the value of a BTC on the open market has fluctuated a lot over the years. Recently, the value of one BTC has been in the 600–650 USD range.[3] Finally, the transaction fees won't play much of a role until next lecture, so you can ignore them for now. We'll talk shortly about the who and the how of authorizing transactions.

A transaction is *valid* if:

1. It has been cryptographically signed by all of the senders. (This can be verified using the senders' public keys.)

2. The senders really do own the coins being transferred.

The second criterion is also easy to verify, given how Bitcoin works. Specifically, transactions are broadcast to all other users (through a peer-to-peer network), and all users keep track of all transactions that have ever been authorized. Thus, everyone knows everyone's current balance, and whether they're in a position to make the specified transfer. The record of all transactions that have been authorized so far is called the *ledger*, and it is the sun around which the Bitcoin world orbits.

Many questions come to mind. For example, how do transactions get authorized and added to the ledger? (Traditionally, this would done by a centralized entity like a bank.) Also, how do Bitcoins get created in the first place? (Traditionally, money is printed by the government.) Amazingly, Bitcoin addresses both these issues in one fell swoop.

2 Bitcoin's Blockchain Protocol

2.1 Blocks

Transactions are added to the ledger in groups (rather than one-by-one), known as *blocks*. Specifically, a block has the following ingredients:

1. One or more transactions.

2. A hash of the previous block.

3. A nonce. (I.e., a bunch of bits that can be set arbitrarily.)

[2]This means that the same entity can correspond to multiple different Bitcoin users. The ability to create multiple identities in a system is often a big problem (see the next lecture), but remarkably, it's no problem for Bitcoin (Section 2.6).

[3]Users can also transfer fractions of Bitcoins—any multiple of 10^{-9} BTC (called *Satoshis*) is allowed.

Blocks typically have on the order of 1000–2000 transactions (there is a 1 MB cap on the block size). The second ingredient imposes a natural linked list-type structure on the ledger, with the predecessor of a block b_2 being the block b_1 whose hash matches the hash stored in b_2. See Figure 1; this is the *blockchain*.

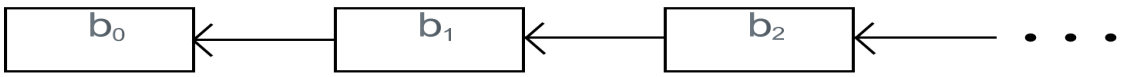

Figure 1: Bitcoin's blockchain. Each block contains a hash of the previous block, imposing a linked-list-type structure on the ledger.

2.2 The High-Level Idea

How do new blocks get added to the blockchain? Who can do it? Why should they bother? How can we make sure that everybody agrees on the contents of the blockchain? The two-prong solution to these questions is brilliant in its elegance.

1. Any user can authorize a block. Bitcoin incentivizes users to do authorizations through explicit monetary rewards (in BTC, naturally).

2. To avoid anarchy, authorizing a new block of transactions involves a *proof of work*, meaning that the authorizer has to solve a computationally difficult puzzle.[4]

2.3 Computationally Difficult Puzzles

Now for the details, beginning with the second point. An important definition: a block b (consisting of transactions, the hash of the previous block, and the nonce) is *valid* if $h(b)$ is sufficiently close to 0, where h is a pre-agreed upon hash function (currently, SHA-256, so $h(b)$ is 256 bits). Thus finding a valid block more or less involves inverting a one-way/cryptographic hash function. By "sufficiently close to 0," we mean that the leading ℓ bits of $h(b)$ should all be 0, where ℓ is a parameter. Note that ℓ provides a knob to control the difficulty of the problem: the bigger the choice of ℓ, the harder the problem.[5] How is ℓ chosen? To keep the rate of valid block creation roughly constant, averaging around one block every ten minutes. The parameter ℓ is re-calibrated roughly every two weeks, based on the average time between blocks during this window. (So if the average time between blocks has dropped to 9 minutes,

[4]Proofs of work have been around for awhile—e.g., in the spam-fighting proposal in [1] (from 1992, which basically predates spam!)—but Bitcoin has emerged as the real killer application of the idea.

[5]Inverting a one-way function is one of the few computational problems where the difficulty can be tightly controlled through a natural parameter. For example, how you would define an equally tunable family of instances of the satisfiability problem?

it's time to increase ℓ.) Why 10 minutes? Because to maintain the property that all miners are on the same page about what the current blockchain is, it seems essential to have block creation happening on a time scale slower than the latencies in the peer-to-peer network. (So 5 minutes between blocks would probably still work fine, but 1 minute might be asking for trouble.) Currently, ℓ is roughly 70.

2.4 Block Rewards and Bitcoin Mining

Bitcoin mining refers to the process of finding new valid blocks. The intended behavior for a bitcoin miner is: choose a subset of the outstanding transactions that it knows about (e.g., the ones with the highest transaction fees), insert the hash of the current last block of the blockchain, arbitrarily set the bits in the nonce, and hope that the resulting block b is valid. Since h is a cryptographic hash function, the accepted belief is that there is no algorithm for finding a valid block that is smarter or faster than random guessing or exhaustive search. (This has not been proved formally, but is consistent with the current state-of-the-art.) Assuming that SHA-256 acts like a random function, for any given block b, the probability that b is valid is $2^{-\ell}$. This means that the expected number of tries necessary to find a valid block is 2^{ℓ}. In practice, Bitcoin miners typically search for a valid block by fixing the set of transactions and varying the nonce until the block becomes valid.[6,7] When a new valid block has been found, the miner is supposed to immediately broadcast it across the entire peer-to-peer network, so that it gets appended to the blockchain (and the miner can collect its reward, see below). If someone else announces a new valid block first, then the miner restarts this procedure, now using only transactions not already authorized by the new block, and using the hash of the new block.

The reward that a miner gets for adding a new (valid) block to the blockchain has two ingredients.

1. A flat reward that does not depend on the contents of the block (other than it being valid). When Bitcoin debuted this reward was 50 BTC, but the protocol dictates that this amount gets cut in half every four years. It was cut to 25 BTC in 2012, and to 12.5 BTC just this past summer. At current exchange rates, this amounts to an 8K or 9K USD reward per block.

2. The sum of the transaction fees of the transactions in the block. Currently, transaction fees are non-zero but typically constitute only a few percent of the overall reward.[8]

[6] 2^{70} is a really big number! A bitcoin miner who only uses CPUs to search for valid blocks would not find one for hundreds of thousands of years. Using lots of GPUs would bring the expected wait time down to the hundreds of years. These days, the only viable way of bitcoin mining is via ASICs (application-specific integrated circuits), meaning hardware that is built specifically to search for valid blocks and do nothing else.

[7] The nonce has enough bits that there will be a "magic nonce" that renders the block valid—but such nonces will be hard to find.

[8] Next lecture we'll speculate about incentive issues when transaction fees are large.

The transaction fees are just a transfer of bitcoins between users (from the transaction originators to the authorizer). The flat reward, however, represents newly minted bitcoins. *This is the only way that new money gets printed.* Currently, there are roughly 15 million BTCs out there. Because the rate of bitcoin creation is decaying exponentially with time, only a finite number (roughly 21 million) will ever be created (unless the protocol is changed).

2.5 Forks

Once in a while, two different bitcoin miners will discover valid blocks at roughly the same time. This results in a *fork* in the blockchain (Figure 2), where two valid blocks, each with its own set of transactions, point to the same previous block. There needs to be a mechanism for deciding which branch of the fork is the "right" one, for two reasons: (i) so that everybody knows which transactions have been authorized; and (ii) so that bitcoin miners know which block they should be trying to extend.

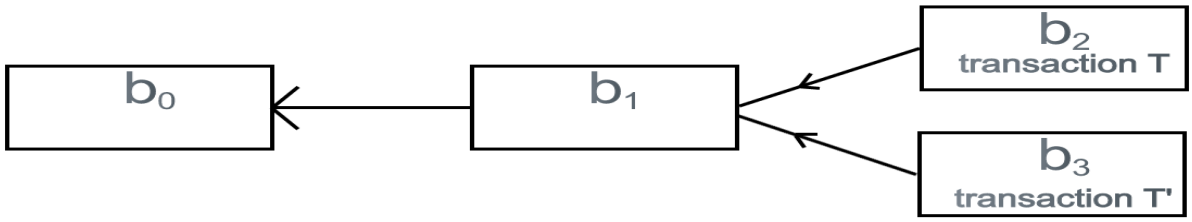

Figure 2: Fork in the blockchain, caused by two new blocks being discovered by different miners at roughly the same time.

The Bitcoin protocol specifies the intended behavior when there's a fork: a user should regard the longest branch as the valid one, breaking ties according to the block that it heard about first. When there is a fork as in Figure 2, it is completely possible that different users will have different opinions about which branch is the valid one (user 1 may have heard about b_2 before b_3, while user 2 heard about b_3 before b_2).

Eventually, some bitcoin miner will authorize a new block, which extends only one of the branches (depending on which hash the miner put in the new block). If there were previously branches with equal length, then this new block will break the tie and create a chain longer than any other. At this point, all users will again have a consistent view of the blockchain (as the longest chain). When this happens, blocks not on this longest chain are "orphaned," and the transactions in them are not regarded as authorized. (Some of them may be authorized instead by a block on the longest chain.) Similarly, the creators of these orphaned blocks do not get any reward for them.

Blocks will occasionally get orphaned even when all miners are obediently following the protocol. Thus, one should not regard a transaction as authorized until it has been included in a block on the blockchain, *and also* been extended by another block (Figure 3). For example, when a user buys some goods using a Bitcoin transaction, the merchant should not ship the goods until the transaction shows up and has been extended in the blockchain.

More conservative sellers can choose to wait until some number $k \geq 1$ of blocks have been appended to the block containing the relevant transaction.

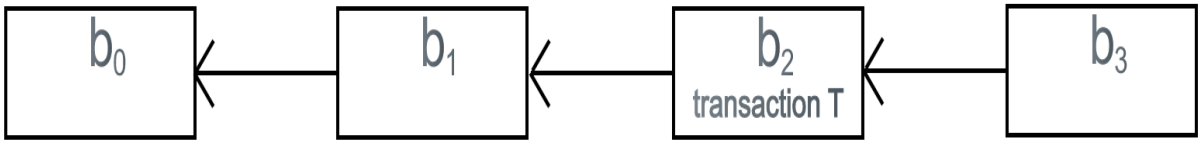

Figure 3: The transaction T between the user and the merchant is in block b_2. The merchant, for safety, should wait for the next block b_3 to be added to the blockchain before shipping the goods.

2.6 Sybil Attacks

Bitcoin users are identified with public keys. It's not hard to generate many public keys, so many Bitcoin "users" might actually correspond to the same person. Deliberately creating multiple identities in a system is often called a *Sybil attack*.[9] Sybil attacks plague many systems (see the next lecture), but remarkably, they cause no issues in Bitcoin. Your influence in Bitcoin is determined entirely by the amount of computational power that you wield—the number of identities is irrelevant.

3 Incentive Issues

There are many ways that a bitcoin miner could conceivably deviate from the intended behavior, and we explore some of these next. At the outset, we should say that there have been little to no sightings of any of these attacks "in the wild." The usual explanation for this is that any major attack on Bitcoin would immediately devalue the currency, a losing situation for everyone (including the attacker). Still, it's important to think through what kinds of attacks we might see in the future. Some of the more recently identified attacks are fairly subtle and hard-to-detect, and may become increasingly prevalent over time.

3.1 The Double-Spend Attack

The first type of deviation we'll look at is when miners *deliberately create forks.* This is simple to do: when searching for a valid block, just insert the hash of a block on the blockchain that is not the last block.

To see why a miner might want to create a fork, suppose in the transaction T, Alice transfers some bitcoins to Bob. Suppose this transaction gets added to the blockchain as part of block b_1. Per the discussion above, Bob only ships the purchased goods to Alice once

[9]Named after the 1973 book *Sybil* about a woman with what was then called a "multiple personality disorder."

another block b_2 has been appended to b_1. When Alice has the goods, she could try the following attack: try to find a valid block b_3 extending b_0, another block b_4 extending b_3, and a third block b_5 extending b_4. Alice does not put transaction T in any of these blocks. If Alice successfully creates these three blocks before any other miner extends b_2, then she rips off Bob: b_1 and b_2 are orphaned and Alice's payment to Bob gets canceled, while the goods have already been sent.[10] This attack is sometimes called the *double-spend attack*, especially in the case where Alice puts a payment T' to Carol in the block b_3, promising the same coins to Carol that she already promised to Bob. Since bits are easily copied, every digital currency must address double-spending attacks.

The probability that Alice succeeds in her double-spend attack depends on how much computational power she has. Suppose that of all the computational cycles being devoted to bitcoin mining, Alice controls an α fraction. The fraction α is sometimes called Alice's *mining power*. Since finding valid blocks just boils down to random guessing or exhaustive search, the probability that Alice is the one who finds the next valid block is well-approximated by α. Finding three blocks in a row before anyone else, as needed in the double-spend attack above, happens with probability only α^3. More generally, if Bob waits for $k \geq 1$ blocks to be appended to b_1 before shipping the goods, then the probability that a double-spend attack succeeds is only α^{k+2}.

How big is α? If just a solo miner, then not very big. However many miners participate in *mining pools*, where the miners join forces, all working on the same puzzle, and sharing the rewards of finding a valid block among the pool members. The reward is shared proportionally, according to the amount of computation contributed by each member of the pool.[11] And big mining pools can control a significant fraction of the total computational power (e.g. $\alpha = .3$).

3.2 The 51% Attack

On the surface, our discussion of a double-spend attack above would seek to apply even if $\alpha = .51$—even here the success probability is only $\approx \frac{1}{8}$. But when $\alpha > \frac{1}{2}$, a more patient strategy is guaranteed to succeed: Alice just continues to extend her own chain (b_3, b_4, b_5, ...) until it is the longest chain. Since on average Alice creates more than every other block, her chain will eventually overtake any other chain.

More generally, Bitcoin is not intended to function when a single entity controls more than 50% of the computational power. Such an entity can effectively act as a centralized authority, defeating the whole point of Bitcoin. For example, while such an entity cannot outright steal bitcoins from another user's account (because of the cryptographic protections), it can freeze the assets of any user that it wants, by forcing any blocks involving that user to be

[10] Assuming all other miners behave as intended, appending only two blocks b_3 and b_4 is not enough. The reason is that other miners will have heard about b_2 first, and ties are broken in favor of blocks heard about earlier.

[11] Pool members prove their computational contribution by reporting all of the "partial solutions" that they found—e.g., blocks b where $h(b)$ has 50 leading zeroes (rather than the requisite 70).

orphaned.[12] In general, it is only interesting to study Bitcoin when no one controls more than 50% of the overall computational power.

3.3 Selfish Mining

All of our previous attacks involved deliberate forking. Next we discuss a different type of deviation from the intended behavior: *block withholding*. Recall that a miner is supposed to announce a valid block to everyone as soon as she finds one; but if she desires, she can opt to keep it a secret.

What is the incentive for Alice to withhold a block and forego the corresponding reward? The intuition is that Alice can trick all of the other miners into working on the wrong computational problem (extending the last publicly announced block, not Alice's secret block). Meanwhile, Alice can work privately on extending her own block. This trick boosts Alice's fraction of the (useful) computation being done, and hence has the potential to boost her expected reward.

In detail, here's the *selfish mining* strategy. Suppose the last block of the current blockchain is b_0, and Alice just discovered a new valid block s_1 (which she will withhold from the others).

Selfish Mining

1. Work privately to find a valid block s_2 that extends s_1. If some other miner announces a new valid block b_1 (extending b_0), then give up and start over.[13]

2. If Alice finds s_2 (extending s_1) before any other miner finds b_1 (extending b_0), then Alice continues to mine her secret chain $s_1, s_2, \ldots, s_k$ until her "lead" over the public blockchain $b_1, b_2, \ldots, b_\ell$ drops to 1 (i.e., $\ell = k-1$).

3. Announce her entire secret chain $s_1, \ldots, s_k$.

For example, suppose Alice gets lucky and finds s_2 and s_3, and only then does some other miner find b_1. Alice continues to try to extend the end s_3 of her secret chain (with a lead of 2, this is still safe). If some other miner finds b_2 first, however, then Alice cashes in her chips and announces s_1, s_2, and s_3.

Assume that all miners other than Alice behave as intended. Would selfish mining give Alice a higher expected reward than honest mining? The answer is not obvious, but can be

[12] One mining pool, Ghash.io, once reached roughly 50% of the total computational power. As the pool had no interest in attacking Bitcoin (this would devalue the currency), it took steps to bring the membership back down to below 40%.

[13] This is the bad scenario for the selfish miner—she could have collected a reward for the block s_1, but once b_1 is announced she can no longer collect any reward for it. (Recall miners are supposed to break ties in favor of the block announced earlier, so announcing s_1 now would not help.)

obtained through a calculation.[14] And the answer is interesting:

Theorem 3.1 ([2]) *If Alice's mining power α is bigger than $\frac{1}{3}$, and all other miners are honest, then selfish mining yields greater expected reward than honest mining.*

Interestingly, the original white paper on Bitcoin [3], by a mysterious individual or team known only as Satoshi Nakamoto, seems to suggest that there should be no incentive issues provided no miner has more than 50% of the overall computational power. Presumably Nakamoto had in mind attacks based on forking (like the double-spend attack), rather than attacks based on block withholding.

Theorem 3.1 effectively says that all miners being honest is not an "equilibrium," when at least one miner controls more than a third of the overall computational power. So what are the equilibria? That seem like a very difficult question.

References

[1] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference (CRYPTO)*, pages 139–147, 1992.

[2] I. Eyal and E. Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Proceedings of the Eighteenth International Conference on Financial Cryptography and Data Security*, 2014.

[3] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Unpublished white paper, 2008.

[14] Basically by computing the stationary distribution of a suitable Markov chain (where states correspond to different-sized "leads") and then the expected rewards.