How the Bitcoin protocol actually works

by Michael Nielsen on December 6, 2013

Many thousands of articles have been written purporting to explain Bitcoin, the online, peer-to-peer currency. Most of those articles give a hand-wavy account of the underlying cryptographic protocol, omitting many details. Even those articles which delve deeper often gloss over crucial points. My aim in this post is to explain the major ideas behind the Bitcoin protocol in a clear, easily comprehensible way. We'll start from first principles, build up to a broad theoretical understanding of how the protocol works, and then dig down into the nitty-gritty, examining the raw data in a Bitcoin transaction.

Understanding the protocol in this detailed way is hard work. It is tempting instead to take Bitcoin as given, and to engage in speculation about how to get rich with Bitcoin, whether Bitcoin is a bubble, whether Bitcoin might one day mean the end of taxation, and so on. That's fun, but severely limits your understanding. Understanding the details of the Bitcoin protocol opens up otherwise inaccessible vistas. In particular, it's the basis for understanding Bitcoin's built-in scripting language, which makes it possible to use Bitcoin to create new types of financial instruments, such as **smart contracts**. New financial instruments can, in turn, be used to create new markets and to enable new forms of collective human behaviour. Talk about fun!

I'll describe Bitcoin scripting and concepts such as smart contracts in future posts. This post concentrates on explaining the nuts-and-bolts of the Bitcoin protocol. To understand the post, you need to be comfortable with **public key cryptography**, and with the closely related idea of **digital signatures**. I'll also assume you're familiar with **cryptographic hashing**. None of this is especially difficult. The basic ideas can be taught in freshman university mathematics or computer science classes. The ideas are beautiful, so if you're not familiar with them, I recommend taking a few hours to get familiar.

It may seem surprising that Bitcoin's basis is cryptography. Isn't Bitcoin a currency, not a way of sending secret messages? In fact, the problems Bitcoin needs to solve are largely about securing transactions — making sure people can't steal from one another, or impersonate one another, and so on. In the world of atoms we achieve security with devices such as locks, safes, signatures, and bank vaults. In the world of bits we achieve this kind of security with cryptography. And that's why Bitcoin is at heart a cryptographic protocol.

My strategy in the post is to build Bitcoin up in stages. I'll begin by explaining a very simple digital currency, based on ideas that are almost obvious. We'll call that currency *Infocoin*, to distinguish it from Bitcoin. Of course, our first version of Infocoin will have many deficiencies, and so we'll go through several iterations of Infocoin, with each iteration introducing just one or two simple new ideas. After several such iterations, we'll arrive at the full Bitcoin protocol. We will have reinvented Bitcoin!

This strategy is slower than if I explained the entire Bitcoin protocol in one shot. But while you can understand the mechanics of Bitcoin through such a one-shot explanation, it would be difficult to understand *why* Bitcoin is designed the way it is. The advantage of the slower iterative explanation is that it gives us a much sharper understanding of each element of Bitcoin.

Finally, I should mention that I'm a relative newcomer to Bitcoin. I've been following it loosely since 2011 (and cryptocurrencies since the late 1990s), but only got seriously into the details of the Bitcoin protocol earlier this year. So I'd certainly appreciate corrections of any misapprehensions on my part. Also in the post I've included a number of "problems for the author" – notes to myself about questions that came up during the writing. You may find these interesting, but you can also skip them entirely without losing track of the main text.

First steps: a signed letter of intent

Follow Michael

Subscribe to this blog
Michael's main blog
Follow Michael on Twitter

Other places on the web

GitHub
Delicious
Delicious research links for book project
Delicious research links for last book
Google Plus

Books

Reinventing Discovery: The New Era of Networked Science (Errata)

Quantum Computation and Quantum Information

Other projects

Essays
TEDxWaterloo video about open science
Quantum computing for the determined
Polymath wiki

Recent Posts

How the backpropagation algorithm works
Reinventing Explanation
How the Bitcoin protocol actually works
Neural Networks and Deep Learning: first chapter now live
Why Bloom filters work the way they do

Recent Comments

Rich on How the Bitcoin protocol actually works
Kaitlyn on If correlation doesn't imply causation, then what does?
Kaitlyn on If correlation doesn't imply causation, then what does?
What I learned at Build Peace (by Jonathan Stray) | Build Peace on If correlation doesn't imply causation, then what does?
What I learned at Build Peace (Jonathan Stray) | Build Peace on If correlation doesn't imply causation, then what does?

Linklog

Documentaries

So how can we design a digital currency?

On the face of it, a digital currency sounds impossible. Suppose some person – let's call her Alice – has some digital money which she wants to spend. If Alice can use a string of bits as money, how can we prevent her from using the same bit string over and over, thus minting an infinite supply of money? Or, if we can somehow solve that problem, how can we prevent someone else forging such a string of bits, and using that to steal from Alice?

These are just two of the many problems that must be overcome in order to use information as money.

As a first version of Infocoin, let's find a way that Alice can use a string of bits as a (very primitive and incomplete) form of money, in a way that gives her at least some protection against forgery. Suppose Alice wants to give another person, Bob, an infocoin. To do this, Alice writes down the message "I, Alice, am giving Bob one infocoin". She then digitally signs the message using a private cryptographic key, and announces the signed string of bits to the entire world.

(By the way, I'm using capitalized "Infocoin" to refer to the protocol and general concept, and lowercase "infocoin" to refer to specific denominations of the currency. A similar useage is common, though not universal, in the Bitcoin world.)

This isn't terribly impressive as a prototype digital currency! But it does have some virtues. Anyone in the world (including Bob) can use Alice's public key to verify that Alice really was the person who signed the message "I, Alice, am giving Bob one infocoin". No-one else could have created that bit string, and so Alice can't turn around and say "No, I didn't mean to give Bob an infocoin". So the protocol establishes that Alice truly intends to give Bob one infocoin. The same fact – no-one else could compose such a signed message – also gives Alice some limited protection from forgery. Of course, *after* Alice has published her message it's possible for other people to duplicate the message, so in that sense forgery is possible. But it's not possible from scratch. These two properties – establishment of intent on Alice's part, and the limited protection from forgery – are genuinely notable features of this protocol.

I haven't (quite) said exactly what digital money *is* in this protocol. To make this explicit: it's just the message itself, i.e., the string of bits representing the digitally signed message "I, Alice, am giving Bob one infocoin". Later protocols will be similar, in that all our forms of digital money will be just more and more elaborate messages [1].

Using serial numbers to make coins uniquely identifiable

A problem with the first version of Infocoin is that Alice could keep sending Bob the same signed message over and over. Suppose Bob receives ten copies of the signed message "I, Alice, am giving Bob one infocoin". Does that mean Alice sent Bob ten *different* infocoins? Was her message accidentally duplicated? Perhaps she was trying to trick Bob into believing that she had given him ten different infocoins, when the message only proves to the world that she intends to transfer one infocoin.

What we'd like is a way of making infocoins unique. They need a label or serial number. Alice would sign the message "I, Alice, am giving Bob one infocoin, with serial number 8740348". Then, later, Alice could sign the message "I, Alice, am giving Bob one infocoin, with serial number 8770431", and Bob (and everyone else) would know that a different infocoin was being transferred.

To make this scheme work we need a trusted source of serial numbers for the infocoins. One way to create such a source is to introduce a *bank*. This bank would provide serial numbers for infocoins, keep track of who has which infocoins, and verify that transactions really are legitimate,

In more detail, let's suppose Alice goes into the bank, and says "I want to withdraw one infocoin from my account". The bank reduces her account balance by one infocoin, and assigns her a new, never-before used serial number, let's say 1234567. Then, when Alice wants to transfer her infocoin to Bob, she signs the message "I, Alice, am giving Bob one infocoin, with serial number 1234567". But Bob doesn't just accept the infocoin. Instead, he contacts the bank, and verifies that: (a) the infocoin with that serial number belongs to Alice; and (b) Alice hasn't already spent the infocoin. If both those things are true, then Bob tells the bank he wants to accept the infocoin, and the bank updates their records to show that the infocoin with that serial number is now in Bob's possession, and no longer belongs to Alice.

Making everyone collectively the bank

This last solution looks pretty promising. However, it turns out that we can do something much more ambitious. We can eliminate the bank entirely from the protocol. This changes the nature of the currency considerably. It means that there is no longer any single organization in charge of the currency. And when you think about the enormous power a central bank has – control over the money supply – that's a pretty huge change.

The idea is to make it so *everyone* (collectively) is the bank. In particular, we'll assume that everyone using Infocoin keeps a complete record of which infocoins belong to which person. You can think of this as a shared public ledger showing all Infocoin transactions. We'll call this ledger the *block chain*, since that's what the complete record will be called in Bitcoin, once we get to it.

Now, suppose Alice wants to transfer an infocoin to Bob. She signs the message "I, Alice, am giving Bob one infocoin, with serial number 1234567", and gives the signed message to Bob. Bob can use his copy of the block chain to check that, indeed, the infocoin is Alice's to give. If that checks out then he broadcasts both Alice's message and his acceptance of the transaction to the entire network, and everyone updates their copy of the block chain.

We still have the "where do serial number come from" problem, but that turns out to be pretty easy to solve, and so I will defer it to later, in the discussion of Bitcoin. A more challenging problem is that this protocol allows Alice to cheat by double spending her infocoin. She sends the signed message "I, Alice, am giving Bob one infocoin, with serial number 1234567″ to Bob, and the message"I, Alice, am giving Charlie one infocoin, with [the same] serial number 1234567" to Charlie. Both Bob and Charlie use their copy of the block chain to verify that the infocoin is Alice's to spend. Provided they do this verification at nearly the same time (before they've had a chance to hear from one another), both will find that, yes, the block chain shows the coin belongs to Alice. And so they will both accept the transaction, and also broadcast their acceptance of the transaction. Now there's a problem. How should other people update their block chains? There may be no easy way to achieve a consistent shared ledger of transactions. And even if everyone can agree on a consistent way to update their block chains, there is still the problem that either Bob or Charlie will be cheated.

At first glance double spending seems difficult for Alice to pull off. After all, if Alice sends the message first to Bob, then Bob can verify the message, and tell everyone else in the network (including Charlie) to update their block chain. Once that has happened, Charlie would no longer be fooled by Alice. So there is most likely only a brief period of time in which Alice can double spend. However, it's obviously undesirable to have any such a period of time. Worse, there are techniques Alice could use to make that period longer. She could, for example, use network traffic analysis to find times when Bob and Charlie are likely to have a lot of latency in communication. Or perhaps she could do something to deliberately disrupt their communications. If she can slow communication even a little that makes her task of double spending much easier.

How can we address the problem of double spending? The obvious solution is that when Alice sends Bob an infocoin, Bob shouldn't try to verify the transaction alone. Rather, he should broadcast the possible transaction to the entire network of Infocoin users, and ask them to help determine whether the transaction is legitimate. If they collectively decide that the transaction is okay, then Bob can accept the infocoin, and everyone will update their block chain. This type of protocol can help prevent double spending, since if Alice tries to spend her infocoin with both Bob and Charlie, other people on the network will notice, and network users will tell both Bob and Charlie that there is a problem with the transaction, and the transaction shouldn't go through.

In more detail, let's suppose Alice wants to give Bob an infocoin. As before, she signs the message "I, Alice, am giving Bob one infocoin, with serial number 1234567", and gives the signed message to Bob. Also as before, Bob does a sanity check, using his copy of the block chain to check that, indeed, the coin currently belongs to Alice. But at that point the protocol is modified. Bob doesn't just go ahead and accept the transaction. Instead, he broadcasts Alice's message to the entire network. Other members of the network check to see whether Alice owns that infocoin. If so, they broadcast the message "Yes, Alice owns infocoin 1234567, it can now be transferred to Bob." Once enough people have broadcast that message, everyone updates their block chain to show that infocoin 1234567 now belongs to Bob, and the transaction is complete.

This protocol has many imprecise elements at present. For instance, what does it mean to say "once enough people have broadcast that message"? What exactly does "enough" mean here? It can't mean everyone in the network, since we don't *a priori* know who is on the Infocoin network. For the same reason, it can't mean some fixed fraction of users in the network. We won't try to make these ideas precise right now. Instead, in the next section I'll point out a serious problem with the approach as described. Fixing that problem will at the same time have the pleasant side effect of making the ideas above much more precise.

Proof-of-work

Suppose Alice wants to double spend in the network-based protocol I just described. She could do this by taking over the Infocoin network. Let's suppose she uses an automated system to set up a large number of separate identities, let's say a billion, on the Infocoin network. As before, she tries to double spend the same infocoin with both Bob and Charlie. But when Bob and Charlie ask the network to validate their respective transactions, Alice's sock puppet identities swamp the network, announcing to Bob that they've validated his transaction, and to Charlie that they've validated his transaction, possibly fooling one or both into accepting the transaction.

There's a clever way of avoiding this problem, using an idea known as *proof-of-work*. The idea is counterintuitive and involves a combination of two ideas: (1) to (artificially) make it *computationally costly* for network users to validate transactions; and (2) to *reward* them for trying to help validate transactions. The reward is used so that people on the network will try to help validate transactions, even though that's now been made a computationally costly process. The benefit of making it costly to validate transactions is that validation can no longer be influenced by the number of network identities someone controls, but only by the total computational power they can bring to bear on validation. As we'll see, with some clever design we can make it so a cheater would need enormous computational resources to cheat, making it impractical.

That's the gist of proof-of-work. But to really understand proof-of-work, we need to go through the details.

Suppose Alice broadcasts to the network the news that "I, Alice, am giving Bob one infocoin, with serial number 1234567".

As other people on the network hear that message, each adds it to a queue of pending transactions that they've been told about, but which haven't yet been approved by the network. For instance, another network user named David might have the following queue of pending transactions:

I, Tom, am giving Sue one infocoin, with serial number 1201174.

I, Sydney, am giving Cynthia one infocoin, with serial number 1295618.

I, Alice, am giving Bob one infocoin, with serial number 1234567.

David checks his copy of the block chain, and can see that each transaction is valid. He would like to help out by broadcasting news of that validity to the entire network.

However, before doing that, as part of the validation protocol David is required to solve a hard computational puzzle – the proof-of-work. Without the solution to that puzzle, the rest of the network won't accept his validation of the transaction.

What puzzle does David need to solve? To explain that, let h be a fixed hash function known by everyone in the network – it's built into the protocol. Bitcoin uses the well-known **SHA-256** hash function, but any cryptographically secure hash function will do. Let's give David's queue of pending transactions a label, l, just so it's got a name we can refer to. Suppose David appends a number x (called the *nonce*) to l and hashes the combination. For example, if we use $l =$ "Hello, world!" (obviously this is not a list of transactions, just a string used for illustrative purposes) and the nonce $x = 0$ **then** (output is in hexadecimal)

```
h("Hello, world!0") =
  1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
```

The puzzle David has to solve – the proof-of-work – is to find a nonce x such that when we append x to l and hash the combination the output hash begins with a long run of zeroes. The puzzle can be made more or less difficult by varying the number of zeroes required to solve the puzzle. A relatively simple proof-of-work puzzle might require just three or four zeroes at the start of the hash, while a more difficult proof-of-work puzzle might require a much longer run of zeros, say 15 consecutive zeroes. In either case, the above attempt to find a suitable nonce, with $x = 0$, is a failure, since the output doesn't begin with any zeroes at all. Trying $x = 1$ doesn't work either:

```
h("Hello, world!1") =
  e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8
```

We can keep trying different values for the nonce, $x = 2, 3, \ldots$. Finally, at $x = 4250$ we obtain:

```
h("Hello, world!4250") =
  0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9
```

This nonce gives us a string of four zeroes at the beginning of the output of the hash. This will be enough to solve a simple proof-of-work puzzle, but not enough to solve a more difficult proof-of-work puzzle.

What makes this puzzle hard to solve is the fact that the output from a cryptographic hash function behaves like a random number: change the input even a tiny bit and the output from the hash function changes completely, in a way that's hard to predict. So if we want the output hash value to begin with 10 zeroes, say, then David will need, on average, to try $16^{10} \approx 10^{12}$ different values for x before he finds a suitable nonce. That's a pretty challenging task, requiring lots of computational power.

Obviously, it's possible to make this puzzle more or less difficult to solve by requiring more or fewer zeroes in the output from the hash function. In fact, the Bitcoin protocol gets quite a fine level of control over the difficulty of the puzzle, by using a slight variation on the proof-of-work puzzle described above. Instead of requiring leading zeroes, the Bitcoin proof-of-work puzzle requires the hash of a block's header to be lower than or equal to a number known as the **target**. This target is automatically adjusted to ensure that a Bitcoin block takes, on average, about ten minutes to validate.

(In practice there is a sizeable randomness in how long it takes to validate a block – sometimes a new block is validated in just a minute or two, other times it may take 20 minutes or even longer. It's straightforward to modify the Bitcoin protocol so that the time to validation is much more sharply peaked around ten minutes. Instead of solving a single puzzle, we can require that multiple puzzles be solved; with some careful design it is possible to considerably reduce the variance in the time to validate a block of transactions.)

Alright, let's suppose David is lucky and finds a suitable nonce, x. Celebration! (He'll be rewarded for finding the nonce, as described below). He broadcasts the block of transactions he's approving to the network, together with the value for x. Other participants in the Infocoin network can verify that x is a valid solution to the proof-of-work puzzle. And they then update their block chains to include the new block of transactions.

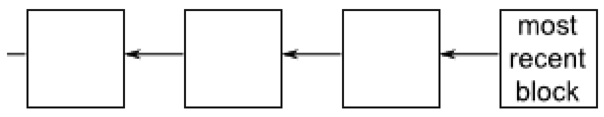
For the proof-of-work idea to have any chance of succeeding, network users need an incentive to help validate transactions. Without such an incentive, they have no reason to expend valuable computational power, merely to help validate other people's transactions. And if network users are not willing to expend that power, then the whole system won't work. The solution to this problem is to reward people who help validate transactions. In particular, suppose we reward whoever successfully validates a block of transactions by crediting them with some infocoins. Provided the infocoin reward is large enough that will give them an incentive to participate in validation.

In the Bitcoin protocol, this validation process is called *mining*. For each block of transactions validated, the successful miner receives a bitcoin reward. Initially, this was set to be a 50 bitcoin reward. But for every 210,000 validated blocks (roughly, once every four years) the reward halves. This has happened just once, to date, and so the current reward for mining a block is 25 bitcoins. This halving in the rate will continue every four years until the year 2140 CE. At that point, the reward for mining will drop below 10^{-8} bitcoins per block. 10^{-8} bitcoins is actually the minimal unit of Bitcoin, and is known as a *satoshi*. So in 2140 CE the total supply of bitcoins will cease to increase. However, that won't eliminate the incentive to help validate transactions. Bitcoin also makes it possible to set aside some currency in a transaction as a *transaction fee*, which goes to the miner who helps validate it. In the early days of Bitcoin transaction fees were mostly set to zero, but as Bitcoin has gained in popularity, transaction fees have gradually risen, and are now a substantial additional incentive on top of the 25 bitcoin reward for mining a block.

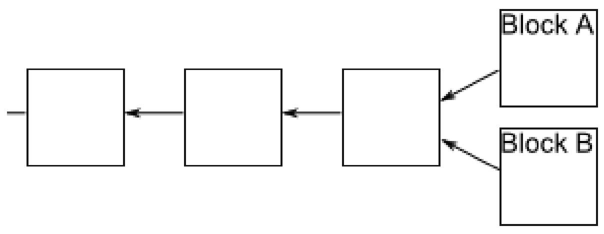
You can think of proof-of-work as a competition to approve transactions. Each entry in the competition costs a little bit of computing power. A miner's chance of winning the competition is (roughly, and with some caveats) equal to the proportion of the total computing power that they control. So, for instance, if a miner controls one percent of the computing power being used to validate Bitcoin transactions, then they have roughly a one percent chance of winning the competition. So provided a lot of computing power is being brought to bear on the competition, a dishonest miner is likely to have only a relatively small chance to corrupt the validation process, unless they expend a huge amount of computing resources.

Of course, while it's encouraging that a dishonest party has only a relatively small chance to corrupt the block chain, that's not enough to give us confidence in the currency. In particular, we haven't yet conclusively addressed the issue of double spending.

I'll analyse double spending shortly. Before doing that, I want to fill in an important detail in the description of Infocoin. We'd ideally like the Infocoin network to agree upon the *order* in which transactions have occurred. If we don't have such an ordering then at any given moment it may not be clear who owns which infocoins. To help do this we'll require that new blocks always include a pointer to the last block validated in the chain, in addition to the list of transactions in the block. (The pointer is actually just a hash of the previous block). So typically the block chain is just a linear chain of blocks of transactions, one after the other, with later blocks each containing a pointer to the immediately prior block:

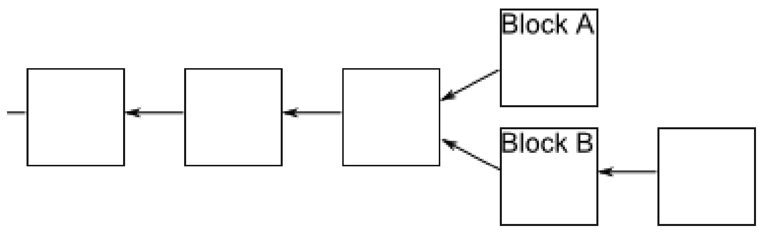


Occasionally, a fork will appear in the block chain. This can happen, for instance, if by chance two miners happen to validate a block of transactions near-simultaneously – both broadcast their newly-validated block out to the network, and some people update their block chain one way, and others update their block chain the other way:



This causes exactly the problem we're trying to avoid – it's no longer clear in what order transactions have occurred, and it may not be clear who owns which infocoins. Fortunately, there's a simple idea that can be used to remove any forks. The rule is this: if a fork occurs, people on the network keep track of both forks. But at any given time, miners only work to extend whichever fork is longest in their copy of the block chain.

Suppose, for example, that we have a fork in which some miners receive block A first, and some miners receive block B first. Those miners who receive block A first will continue mining along that fork, while the others will mine along fork B. Let's suppose that the miners working on fork B are the next to successfully mine a block:



After they receive news that this has happened, the miners working on fork A will notice that fork B is now longer, and will switch to working on that fork. Presto, in short order work on fork A will cease, and everyone will be working on the same linear chain, and block A can be ignored. Of course, any still-pending transactions in A will still be pending in the queues of the miners working on fork B, and so all transactions will eventually be validated.

Likewise, it may be that the miners working on fork A are the first to extend their fork. In that case work on fork B will quickly cease, and again we have a single linear chain.

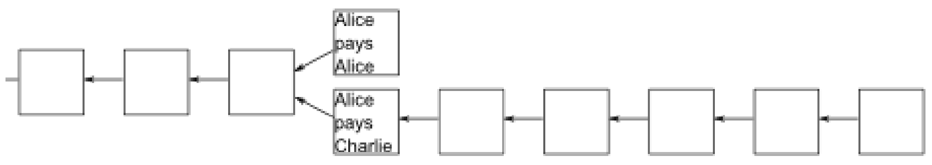
No matter what the outcome, this process ensures that the block chain has an agreed-upon time ordering of the blocks. In Bitcoin proper, a transaction is not considered confirmed until: (1) it is part of a block in the longest fork, and (2) at least 5 blocks follow it in the longest fork. In this case we say that the transaction has "6 confirmations". This gives the network time to come to an agreed-upon the ordering of the blocks. We'll also use this strategy for Infocoin.

With the time-ordering now understood, let's return to think about what happens if a dishonest party tries to double spend. Suppose Alice tries to double spend with Bob and Charlie. One possible approach is for her to try to validate a block that includes both transactions. Assuming she has one percent of the computing power, she will occasionally get lucky and validate the block by solving the proof-of-work.

Unfortunately for Alice, the double spending will be immediately spotted by other people in the Infocoin network and rejected, despite solving the proof-of-work problem. So that's not something we need to worry about.

A more serious problem occurs if she broadcasts two separate transactions in which she spends the same infocoin with Bob and Charlie, respectively. She might, for example, broadcast one transaction to a subset of the miners, and the other transaction to another set of miners, hoping to get both transactions validated in this way. Fortunately, in this case, as we've seen, the network will eventually confirm one of these transactions, but not both. So, for instance, Bob's transaction might ultimately be confirmed, in which case Bob can go ahead confidently. Meanwhile, Charlie will see that his transaction has not been confirmed, and so will decline Alice's offer. So this isn't a problem either. In fact, knowing that this will be the case, there is little reason for Alice to try this in the first place.

An important variant on double spending is if Alice = Bob, i.e., Alice tries to spend a coin with Charlie which she is also "spending" with herself (i.e., giving back to herself). This sounds like it ought to be easy to detect and deal with, but, of course, it's easy on a network to set up multiple identities associated with the same person or organization, so this possibility needs to be considered. In this case, Alice's strategy is to wait until Charlie accepts the infocoin, which happens after the transaction has been confirmed 6 times in the longest chain. She will then attempt to fork the chain before the transaction with Charlie, adding a block which includes a transaction in which she pays herself:



Unfortunately for Alice, it's now very difficult for her to catch up with the longer fork. Other miners won't want to help her out, since they'll be working on the longer fork. And unless Alice is able to solve the proof-of-work at least as fast as everyone else in the network combined – roughly, that means controlling more than fifty percent of the computing power – then she will just keep falling further and further behind. Of course, she might get lucky. We can, for example, imagine a scenario in which Alice controls one percent of the computing power, but happens to get lucky and finds six extra blocks in a row, before the rest of the network has found any extra blocks. In this case, she might be able to get ahead, and get control of the block chain. But this particular event will occur with probability $1/100^6 = 10^{-12}$. A more general analysis along these lines shows that Alice's probability of ever catching up is infinitesimal, unless she is able to solve proof-of-work puzzles at a rate approaching all other miners combined.

Of course, this is not a rigorous security analysis showing that Alice cannot double spend. It's merely an informal plausibility argument. The **original paper** introducing Bitcoin did not, in fact, contain a rigorous security analysis, only informal arguments along the lines I've presented here. The security community is still analysing Bitcoin, and trying to understand possible vulnerabilities. You can see some of this research **listed here**, and I mention a few related problems in the "Problems for the author" below. At this point I think it's fair to say that the jury is still out on how secure Bitcoin is.

The proof-of-work and mining ideas give rise to many questions. How much reward is enough to persuade people to mine? How does the change in supply of infocoins affect the Infocoin economy? Will Infocoin mining end up concentrated in the hands of a few, or many? If it's just a few, doesn't that endanger the security of the system? Presumably transaction fees will eventually equilibriate – won't this introduce an unwanted source of friction, and make small transactions less desirable? These are all great questions, but beyond the scope of this post. I may come back to the questions (in the context of Bitcoin) in a future post. For now, we'll stick to our focus on understanding how the Bitcoin protocol works.

Problems for the author

> I don't understand why double spending can't be prevented in a simpler manner using **two-phase commit**. Suppose Alice tries to double spend an infocoin with both Bob and Charlie. The idea is that Bob and Charlie would each broadcast their respective messages to the Infocoin network, along with a request: "Should I accept this?" They'd then wait some period – perhaps ten minutes – to hear any naysayers who could prove that Alice was trying to double spend. If no such nays are heard (and provided there are no signs of attempts to disrupt the network), they'd then accept the transaction. This protocol needs to be hardened against

network attacks, but it seems to me to be the core of a good alternate idea. How well does this work? What drawbacks and advantages does it have compared to the full Bitcoin protocol?

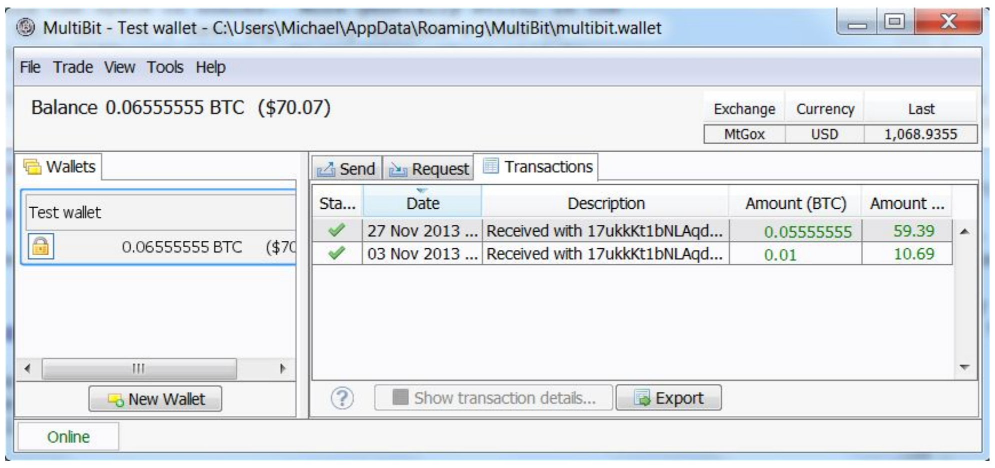
Early in the section I mentioned that there is a natural way of reducing the variance in time required to validate a block of transactions. If that variance is reduced too much, then it creates an interesting attack possibility. Suppose Alice tries to fork the chain in such a way that: (a) one fork starts with a block in which Alice pays herself, while the other fork starts with a block in which Alice pays Bob; (b) both blocks are announced nearly simultaneously, so roughly half the miners will attempt to mine each fork; (c) Alice uses her mining power to try to keep the forks of roughly equal length, mining whichever fork is shorter – this is ordinarily hard to pull off, but becomes significantly easier if the standard deviation of the time-to-validation is much shorter than the network latency; (d) after 5 blocks have been mined on both forks, Alice throws her mining power into making it more likely that Charles's transaction is confirmed; and (e) after confirmation of Charles's transaction, she then throws her computational power into the other fork, and attempts to regain the lead. This balancing strategy will have only a small chance of success. But while the probability is small, it will certainly be much larger than in the standard protocol, with high variance in the time to validate a block. Is there a way of avoiding this problem?

Suppose Bitcoin mining software always explored nonces starting with $x = 0$, then $x = 1, x = 2, \ldots$. If this is done by all (or even just a substantial fraction) of Bitcoin miners then it creates a vulnerability. Namely, it's possible for someone to improve their odds of solving the proof-of-work merely by starting with some other (much larger) nonce. More generally, it may be possible for attackers to exploit any systematic patterns in the way miners explore the space of nonces. More generally still, in the analysis of this section I have implicitly assumed a kind of symmetry between different miners. In practice, there will be asymmetries and a thorough security analysis will need to account for those asymmetries.

Bitcoin

Let's move away from Infocoin, and describe the actual Bitcoin protocol. There are a few new ideas here, but with one exception (discussed below) they're mostly obvious modifications to Infocoin.

To use Bitcoin in practice, you first install a **wallet** program on your computer. To give you a sense of what that means, here's a screenshot of a wallet called **Multbit**. You can see the Bitcoin balance on the left — 0.06555555 Bitcoins, or about 70 dollars at the exchange rate on the day I took this screenshot — and on the right two recent transactions, which deposited those 0.06555555 Bitcoins:



MultiBit - Test wallet - C:\Users\Michael\AppData\Roaming\MultiBit\multibit.wallet

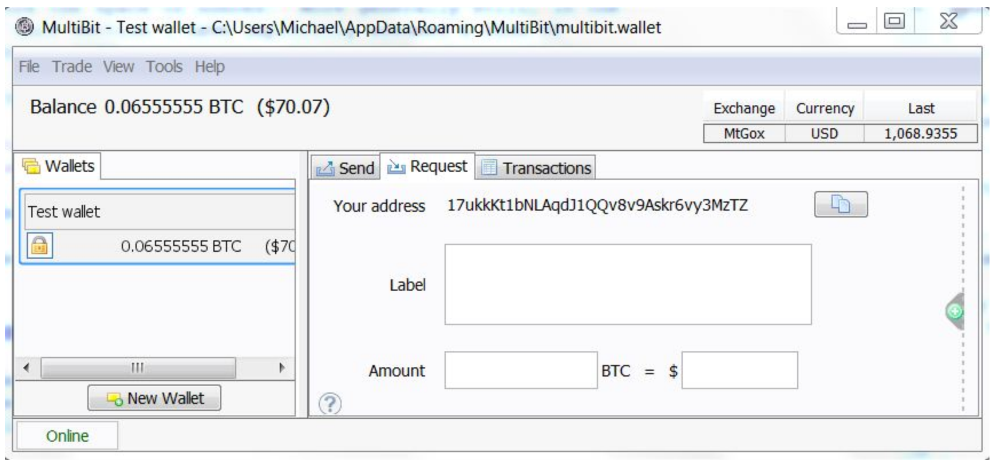
Balance 0.06555555 BTC ($70.07)

Exchange	Currency	Last
MtGox	USD	1,068.9355

Test wallet
0.06555555 BTC ($70

Sta...	Date	Description	Amount (BTC)	Amount ...
✓	27 Nov 2013 ...	Received with 17ukkKt1bNLAqd...	0.05555555	59.39
✓	03 Nov 2013 ...	Received with 17ukkKt1bNLAqd...	0.01	10.69

Suppose you're a merchant who has set up an online store, and you've decided to allow people to pay using Bitcoin. What you do is tell your wallet program to generate a *Bitcoin address*. In response, it will generate a public / private key pair, and then hash the public key to form your Bitcoin address:



MultiBit - Test wallet - C:\Users\Michael\AppData\Roaming\MultiBit\multibit.wallet

Balance 0.06555555 BTC ($70.07)

Exchange	Currency	Last
MtGox	USD	1,068.9355

Test wallet
0.06555555 BTC ($70

Your address 17ukkKt1bNLAqdJ1QQv8v9Askr6vy3MzTZ

Label

Amount BTC = $

You then send your Bitcoin address to the person who wants to buy from you. You could do this in email, or even put the address up publicly on a webpage. This is safe, since the address is merely a hash of your public key, which can safely be known by the world anyway. (I'll return later to the question of why the Bitcoin address is a hash, and not just the public key.)

The person who is going to pay you then generates a *transaction*. Let's take a look at the data from an **actual transaction** transferring 0.31900000 bitcoins. What's shown below is very nearly the raw data. It's changed in three ways: (1) the data has been deserialized; (2) line numbers have been added, for ease of reference; and (3) I've abbreviated various hashes and public keys, just putting in the first six hexadecimal digits of each, when in reality they are much longer. Here's the data:

```
1.  {"hash":"7c4025...",
2.  "ver":1,
3.  "vin_sz":1,
4.  "vout_sz":1,
5.  "lock_time":0,
6.  "size":224,
7.  "in":[
8.    {"prev_out":
9.      {"hash":"2007ae...",
10.       "n":0},
11.    "scriptSig":"304502... 042b2d..."}],
12. "out":[
13.   {"value":"0.31900000",
14.    "scriptPubKey":"OP_DUP OP_HASH160 a7db6f OP_EQUALVERIFY OP_CHECKSIG"}]}
```

Let's go through this, line by line.

Line 1 contains the hash of the remainder of the transaction, `7c4025...`, expressed in hexadecimal. This is used as an identifier for the transaction.

Line 2 tells us that this is a transaction in version 1 of the Bitcoin protocol.

Lines 3 and 4 tell us that the transaction has one input and one output, respectively. I'll talk below about transactions with more inputs and outputs, and why that's useful.

Line 5 contains the value for `lock_time`, which can be used to control when a transaction is finalized. For most Bitcoin transactions being carried out today the `lock_time` is set to 0, which means the transaction is finalized immediately.

Line 6 tells us the size (in bytes) of the transaction. Note that it's not the monetary amount being transferred! That comes later.

Lines 7 through 11 define the input to the transaction. In particular, lines 8 through 10 tell us that the input is to be taken from the output from an earlier transaction, with the given `hash`, which is expressed in hexadecimal as `2007ae...`. The `n=0` tells us it's to be the first output from that transaction; we'll see soon how multiple outputs (and inputs) from a transaction work, so don't worry too much about this for now. Line 11 contains the signature of the person sending the money, `304502...`, followed by a space, and then the corresponding public key, `04b2d...`. Again, these are both in hexadecimal.

One thing to note about the input is that there's nothing explicitly specifying how many bitcoins from the previous transaction should be spent in this transaction. In fact, *all* the bitcoins from the `n=0`th output of the previous transaction are spent. So, for example, if the `n=0`th output of the earlier transaction was 2 bitcoins, then 2 bitcoins will be spent in this transaction. This seems like an inconvenient restriction – like trying to buy bread with a 20 dollar note, and not being able to break the note down. The solution, of course, is to have a mechanism for providing change. This can be done using transactions with multiple inputs and outputs, which we'll discuss in the next section.

Lines 12 through 14 define the output from the transaction. In particular, line 13 tells us the value of the output, 0.319 bitcoins. Line 14 is somewhat complicated. The main thing to note is that the string `a7db6f...` is the Bitcoin address of the intended recipient of the funds (written in hexadecimal). In fact, Line 14 is actually an expression in Bitcoin's scripting language. I'm not going to describe that language in detail in this post, the important thing to take away now is just that `a7db6f...` is the Bitcoin address.

You can now see, by the way, how Bitcoin addresses the question I swept under the rug in the last section: where do Bitcoin serial numbers come from? In fact, the role of the serial number is played by transaction hashes. In the transaction above, for example, the recipient is receiving 0.319 Bitcoins, which come out of the first output of an earlier transaction with hash `2007ae...` (line 9). If you go and look in the block chain for that transaction, you'd see that its output comes from a still earlier transaction. And so on.

There are two clever things about using transaction hashes instead of serial numbers. First, in Bitcoin there's not really any separate, persistent "coins" at all, just a long series of transactions in the block chain. It's a clever idea to realize that you don't need persistent coins, and can just get by with a ledger of transactions. Second, by operating in this way we remove the need for any central authority issuing serial numbers. Instead, the serial numbers can be self-generated, merely by hashing the transaction.

In fact, it's possible to keep following the chain of transactions further back in history. Ultimately, this process must terminate. This can happen in one of two ways. The first possibilitty is that you'll arrive at the very first Bitcoin transaction, contained in the so-called **Genesis block**. This is a special transaction, having no inputs, but a 50 Bitcoin output. In other words, this transaction establishes an initial money supply. The Genesis block is treated separately by Bitcoin clients, and I won't get into the details here, although it's along similar lines to the transaction above. You can see the deserialized raw data **here**, and read about the Genesis block **here**.

The second possibility when you follow a chain of transactions back in time is that eventually you'll arrive at a so-called *coinbase transaction*. With the exception of the Genesis block, every block of transactions in the block chain starts with a special coinbase transaction. This is the transaction rewarding the miner who validated that block of transactions. It uses a similar but not identical format to the transaction above. I won't go through the format in detail, but if you want to see an example, see **here**. You can read a little more about coinbase transactions **here**.

Something I haven't been precise about above is what exactly is being signed by the digital signature in line 11. The obvious thing to do is for the payer to sign the whole transaction (apart from the transaction hash, which, of course, must be generated later). Currently, this is *not* what is done – some pieces of the transaction are omitted. This makes some pieces of the transaction **malleable**, i.e., they can be changed later. However, this malleability does not include the amounts being paid out, senders and recipients, which can't be changed later. I must admit I haven't dug down into the details here. I gather that this malleability is under discussion in the Bitcoin developer community, and there are efforts afoot to reduce or eliminate this malleability.

Transactions with multiple inputs and outputs

In the last section I described how a transaction with a single input and a single output works. In practice, it's often extremely convenient to create Bitcoin transactions with multiple inputs or multiple outputs. I'll talk below about why this can be useful. But first let's take a look at the data from an **actual transaction**:

```
1.  {"hash":"993830...",
2.  "ver":1,
3.  "vin_sz":3,
4.   "vout_sz":2,
5.   "lock_time":0,
6.   "size":552,
7.   "in":[
8.     {"prev_out":{
9.       "hash":"3beabc...",
10.         "n":0},
11.      "scriptSig":"304402... 04c7d2..."},
12.    {"prev_out":{
13.        "hash":"fdae9b...",
14.        "n":0},
15.      "scriptSig":"304502... 026e15..."},
16.    {"prev_out":{
17.        "hash":"20c86b...",
18.        "n":1},
19.      "scriptSig":"304402... 038a52..."}],
20.  "out":[
21.    {"value":"0.01068000",
22.      "scriptPubKey":"OP_DUP OP_HASH160 e8c306... OP_EQUALVERIFY OP_CHECKSIG"}
23.    {"value":"4.00000000",
24.      "scriptPubKey":"OP_DUP OP_HASH160 d644e3... OP_EQUALVERIFY OP_CHECKSIG"}
```

Let's go through the data, line by line. It's very similar to the single-input-single-output transaction, so I'll do this pretty quickly.

Line 1 contains the hash of the remainder of the transaction. This is used as an identifier for the transaction.

Line 2 tells us that this is a transaction in version 1 of the Bitcoin protocol.

Lines 3 and 4 tell us that the transaction has three inputs and two outputs, respectively.

Line 5 contains the `lock_time`. As in the single-input-single-output case this is set to 0, which means the transaction is finalized immediately.

Line 6 tells us the size of the transaction in bytes.

Lines 7 through 19 define a list of the inputs to the transaction. Each corresponds to an output from a previous Bitcoin transaction.

The first input is defined in lines 8 through 11.

In particular, lines 8 through 10 tell us that the input is to be taken from the `n=0`th output from the transaction with `hash 3beabc...`. Line 11 contains the signature, followed by a space, and then the public key of the person sending the bitcoins.

Lines 12 through 15 define the second input, with a similar format to lines 8 through 11. And lines 16 through 19 define the third input.

Lines 20 through 24 define a list containing the two outputs from the transaction.

The first output is defined in lines 21 and 22. Line 21 tells us the value of the output, 0.01068000 bitcoins. As before, line 22 is an expression in Bitcoin's scripting language. The main thing to take away here is that the string `e8c30622...` is the Bitcoin address of the intended recipient of the funds.

The second output is defined lines 23 and 24, with a similar format to the first output.

One apparent oddity in this description is that although each output has a Bitcoin value associated to it, the inputs do not. Of course, the values of the respective inputs can be found by consulting the corresponding outputs in earlier transactions. In a standard Bitcoin transaction, the sum of all the inputs in the transaction must be at least as much as the sum of all the outputs. (The only exception to this principle is the Genesis block, and in coinbase transactions, both of which add to the overall Bitcoin supply.) If the inputs sum up to more than the outputs, then the excess is used as a *transaction fee*. This is paid to whichever miner successfully validates the block which the current transaction is a part of.

That's all there is to multiple-input-multiple-output transactions! They're a pretty simple variation on single-input-single-output-transactions.

One nice application of multiple-input-multiple-output transactions is the idea of *change*. Suppose, for example, that I want to send you 0.15 bitcoins. I can do so by spending money from a previous transaction in which I received 0.2 bitcoins. Of course, I don't want to send you the entire 0.2 bitcoins. The solution is to send you 0.15 bitcoins, and to send 0.05 bitcoins to a Bitcoin address which I own. Those 0.05 bitcoins are the change. Of course, it differs a little from the change you might receive in a store, since change in this case is what you pay yourself. But the broad idea is similar.

Conclusion

That completes a basic description of the main ideas behind Bitcoin. Of course, I've omitted many details – this isn't a formal specification. But I have described the main ideas behind the most common use cases for Bitcoin.

While the rules of Bitcoin are simple and easy to understand, that doesn't mean that it's easy to understand all the consequences of the rules. There is vastly more that could be said about Bitcoin, and I'll investigate some of these issues in future posts.

For now, though, I'll wrap up by addressing a few loose ends.

How anonymous is Bitcoin? Many people claim that Bitcoin can be used anonymously. This claim has led to the formation of marketplaces such as **Silk Road** (and various successors), which specialize in illegal goods. However, the claim that Bitcoin is anonymous is a myth. The block chain is public, meaning that it's possible for anyone to see every Bitcoin transaction ever. Although Bitcoin addresses aren't immediately associated to real-world identities, computer scientists have done a **great deal of work** figuring out how to de-anonymize "anonymous" social networks. The block chain is a marvellous target for these techniques. I will be extremely surprised if the great majority of Bitcoin users are not identified with relatively high confidence and ease in the near future. The confidence won't be high enough to achieve convictions, but will be high enough to identify likely targets. Furthermore, identification will be retrospective, meaning that someone who bought drugs on Silk Road in 2011 will still be identifiable on the basis of the block chain in, say, 2020. These de-anonymization techniques are well known to computer scientists, and, one presumes, therefore to the NSA. I would not be at all surprised if the NSA and other agencies have already de-anonymized many users. It is, in fact, ironic that Bitcoin is often touted as anonymous. It's not. Bitcoin is, instead, perhaps the most open and transparent financial instrument the world has ever seen.

Can you get rich with Bitcoin? Well, maybe. Tim O'Reilly **once said**: "Money is like gas in the car – you need to pay attention or you'll end up on the side of the road – but a well-lived life is not a tour of gas stations!" Much of the interest in Bitcoin comes from people whose life mission seems to be to find a *really big* gas station. I must admit I find this perplexing. What is, I believe, much more interesting and enjoyable is to think of Bitcoin and other cryptocurrencies as a way of enabling new forms of collective behaviour. That's intellectually fascinating, offers marvellous creative possibilities, is socially valuable, and may just also put some money in the bank. But if money in the bank is your primary concern, then I believe that other strategies are much more likely to succeed.

Details I've omitted: Although this post has described the main ideas behind Bitcoin, there are many details I haven't mentioned. One is a nice space-saving trick used by the protocol, based on a data structure known as a **Merkle tree**. It's a detail, but a splendid detail, and worth checking out if fun data structures are your thing. You can get an overview in the **original Bitcoin paper**. Second, I've said little about the **Bitcoin network** – questions like how the network deals with denial of service attacks, how nodes **join and leave the network**, and so on. This is a fascinating topic, but it's also something of a mess of details, and so I've omitted it. You can read more about it at some of the links above.

Bitcoin scripting: In this post I've explained Bitcoin as a form of digital, online money. But this is only a small part of a much bigger and more interesting story. As we've seen, every Bitcoin transaction is associated to a script in the Bitcoin programming language. The scripts we've seen in this post describe simple transactions like "Alice gave Bob 10 bitcoins". But the scripting language can also be used to express far more complicated transactions. To put it another way, Bitcoin is *programmable money*. In later posts I will explain the scripting system, and how it is possible to use Bitcoin scripting as a platform to experiment with all sorts of amazing financial instruments.

*Thanks for reading. Enjoy the essay? You can tip me with Bitcoin (!) at address: 17ukkKt1bNLAqdJ1QQv8v9Askr6vy3MzTZ. You may also enjoy the **first chapter** of my forthcoming book on neural networks and deep learning, and may wish to **follow me on Twitter**.*

Footnote

[1] In the United States the question "Is money a form of speech?" is an important legal question, because of the protection afforded speech under the US Constitution. In my (legally uninformed) opinion digital money may make this issue more complicated. As we'll see, the Bitcoin protocol is really a way of standing up before the rest of the world (or at least the rest of the Bitcoin network) and avowing "I'm going to give such-and-such a number of bitcoins to so-and-so a person" in a way that's extremely difficult to repudiate. At least naively, it looks more like speech than exchanging copper coins, say.

From → Uncategorized

259 Comments

Sumedh **permalink**

Thanks, I was always too lazy to look up BTC in detail. Your article cleared most of my questions.

I wanted to know one thing what if some smart hacker is able to find some vulnerability in the protocol and he uses that to generate new bitcoins for himself. Once that happens then whole confidence in bitcoins would be gone and it would lead to chaos.

Is the above scenario possible?

Bobby **permalink**

Your scenario is possible. Just like any other popular piece of open source software there are incentives for finding exploits, but there are a lot of benevolent hackers examining the code to uncover and fix them.

Michael Nielsen **permalink**

@Bobby: Good point! Yes, that solves much of the problem neatly. My broad point about asymmetries is still true, however. (And is vividly demonstrated by the rise of large mining pools.)

Edit: This is in response to your comment below. I must have clicked on the wrong link when I replied.

achil **permalink**

each block starts with a coinbase transaction which should give bitcoins to the person who solved it. since this transaction if different for any block (each node working on the network has her bitcoin address as the recipient of that transaction), all block in the network are different, so we should'nt (at least not likely) see two blocks with the same hashes.

gwern **permalink**

That bug has actually happened before, but Satoshi/Gavin fixed it before anyone else managed to exploit it. (There have been 2 major live flaws in Bitcoin that I know of: one allowed you to generate billions of bitcoins, and the other allowed you to spend anyone's bitcoins. Neither was exploited before being patched, and there don't seem to have been any big flaws found since.)

Rena **permalink**

Interesting. How did those exploits work?

gwern **permalink**

I don't know the technical details. Might want to look up the CVEs and the patches. From the sound of them, some validation check was omitted and so bad transactions were allowed.

Bobby **permalink**

I believe have the answer to your third question.

The raw block data that each miner is trying to solve contains a generation transaction. That transaction is where their coins are sent if they solve that block. Because miners competing against each other want their coins to be sent to different addresses, and those addresses are hashed together with their nonce, it does not matter if everyone starts their nonce from zero. The added randomness from differing generation transaction addresses prevents each miner from working in the same space as others.

Michael **permalink**

Thanks Bobby. I had wondered about the same question as the author. Your explanation clears it up for me.

JC **permalink**

Moreover the nonces need not be enumerable. If randomly picked from a large enough pool it is unlikely that the same nonce gets picked twice.

Carles **permalink**

Very well written!

Only one thing to add (on another post): when you launch Multibit (or bitcoin-qt, etc.): where does it connect? a list of IPs? DNS? etc. etc.

Pål Driveklepp **permalink**

Bitcoin has 3 methods for finding peers: https://en.bitcoin.it/wiki/Network#Bootstrapping

Benoit **permalink**

Thank you for the great write-up!

You write "I'll return later to the question of why the Bitcoin address is a hash, and not just the public key". Did I miss it? Does it have anything to do with quantum computing?

Michael Nielsen **permalink**

Oops – actually, I had an extended discussion of this question, but deleted it just before I posted. The reason I deleted it is that the discussion was inconclusive. The separation seems to be a fairly arbitrary design decision – there are some minor space and security advantages, but not enough (in my opinion) to justify making the Bitcoin address the hash rather than the public key.

Mark Friedenbach **permalink**

There's a very serious security advantage – the public key is not known until the moment it is spent. That reduces the window during which the private key could be derived and used in a double-spend to about 10 minutes. This has significant ramifications for the safe transition to quantum-proof cryptography, if nothing else.

And space-wise we're talking about saving hundreds of megabytes, soon gigabytes of data from the UTXO set. That's not insignificant.

Michael Nielsen **permalink**

Mark,

You've described the two main points in the material I deleted. To me, both seem like relatively small points. On the first point, many people reuse addresses, so in practice public keys are often widely known. This problem could be reduced if Bitcoin enforced a policy that addresses could only ever be used once, but (a) that's not going to happen anytime soon; (b) it introduces some other practical problems (it's useful to be able to publish an address widely); and (c) it would still leave a window of time when the public key is widely known, but the transaction is not yet confirmed.

On the second point, I haven't computed exactly but I suspect the space savings is less than ten percent of the total transaction size, depending on the type of transaction. That's significant, but it also seems to me that other similar savings could easily have been had, but are not. So it does seem a bit arbitrary. Still, as you point out, it's not insignificant.

(Actually, it occurs to me that it's pretty easy to modify the standard Bitcoin transaction script so only public keys are used. This might make a nice example for my post on Bitcoin scripting.)

David **permalink**

I have read that there is no known algorithm that would allow public keys to be derived from public addresses within a practicable timescale, even with quantum computing. However, the same is not true for deriving private keys from public keys. Thus addresses that have not been used t spend, have benefits in terms of being more QC proof. I recall Vitalik Buterin writing on this topic.

Boldra **permalink**

Could the protocol save even more space by switching from JSON to YAML? It looks like the protocol version is inside the JSON.

peter **permalink**

the json shown in this article is just a representation. the blockchain actually lives in files of binary bytes concatenated together (ie neither jso nor yaml) on each user's computer

Benoit **permalink**

About your first question:

What would be the incentive for non-miners to answer your question?
Why would you trust the answers or lack thereof?

After all, if I understand correctly, when there is no transaction fee set aside, the miners could very well choose to omit transactions from their blocks?

Michael Nielsen **permalink**

On incentive: such could be built into the protocol.

On trusting the answers: if someone claims that they see evidence of double spending, you'd require them to present evidence in the form of a signed transaction. The requirement of a signature makes this hard to forge by a malicious naysayer.

On your last point, yes, this is a very interesting question. At present this all seems to be working okay, but over the long run I suspect will limit the use of Bitcoin for small transactions.

Tom **permalink**

On the last point: I could see the transaction fee being indirectly related to the time required to confirm a transfer. If you want your transfer confirmed quicker, then you have to pay.

On a related note – what happens if blocks are validated out of time order?

MG **permalink**

I'm curious what happens if/when the underlying crypto is either undermined or broken?

Over the years we've seen flaws that reduce the bits (entropy) in many many systems. How would the bitcoin protocol handle, say, a reduction of even 1 bit of difficulty (1 bit reduction == 1/2 as hard to attack) .

Also could someone with very large resources overwhelm the network with bad data? Eg, if china wanted to use some super computers or a bot net to stop bitcoin from operating by adding all sorts of bad data to the block chains?

Michael Nielsen **permalink**

Denial of service type attacks are a real problem. See, e.g:
https://en.bitcoin.it/wiki/weaknesses#Denial_of_Service_.28DoS.29_attacks

On the first question, the answer is, I think: "That's really complicated, and depends on the exact scenario of the break".

cabin permalink

Android had a bug in their random number api that was successfully exploited. Losing a few bits of entropy won't matter, but in this case they lost nearly all of them.

http://thegenesisblock.com/security-vulnerability-in-all-android-bitcoin-wallets/

MG permalink

One additional question, what happens to bitcoins that are "lost" . ie What happens if the FBI refuses to sign over the bitcoins seized from Silk road, or wallets that had some coins but were lost due to hard drive failure (bad backups) or lost passphrases? Or maybe someone dies but the next of kin doesnt know the details?

The comparison is If I drop $20 on the ground or my next of kin finds it under a mattress, they can use it.

Michael Nielsen permalink

Lost bitcoins are just that – gone from the money supply for good, unless someone manages to either (a) recover the keypair; or (b) breaks the underlying crypto.

Mango Cat permalink

That brings up an interesting scenario, on a long time scale there will have to be some allowance made for replacement of the lost coins, or sub-division of the satoshi.

James H permalink

With Bitcoin; losing the private key for good is more like accidentally dropping your coins out of an airplane over the pacific ocean.

The private key is crucial to recovering those coins.

Hasitha N. Liyanage permalink

Looks like we both independently arrived at similar methods of explanation: http://zen.lk/2013/11/28/how-i-finally-understood-bitcoin/

Amos permalink

In the second paragraph of the Bitcoin section, seems it should be 0.06555555, not 0.6555555

Michael Nielsen permalink

Thanks, fixed.

Vidya permalink

Thank you so much !!!! I had wanted an understandable primer on Bitcoin since ages and this was a fabulous read !

Boldra permalink

I'm surprised that bitcoins in a transaction are a decimal string. It looks likely to cause floating point approximation errors.

Have you read about coin-join? It follows on very nicely from what you've described here.

I'm looking forward to the next one

Michael Nielsen permalink

They're not actually a float — as I mention in the article, the minimal unit of Bitcoin is the Satoshi, which is one one hundred millionth of a Bitcoin. So it's really specifying an integer number of Satoshis.

(I haven't checked what type is used in the source code; I'd be interested to know.)

Michael Nielsen permalink

This page says that it is an integer in the original source:

https://en.bitcoin.it/wiki/Proper_Money_Handling_(JSON-RPC)

However, it sounds as though there can occasionally be some floating point / rounding issues with code used to do remote procedure calls over the Bitcoin network.

Uri permalink

Just wanted to say thanks for a really great essay — the explanation was really clear, and totally fascinating.

Jim permalink

Can quantum computers mine bitcoin faster? Does this boil down to how quickly a quantum computer can find a string that has a specified property for SHA-256? For which we have a quadratic speedup, but probably no more?

I understand that commonly used digital signatures and public-key cryptosystems are broken by quantum computers, so there's not much to be said about that.

Michael Nielsen permalink

I haven't thought much about it. With that said, I'm pretty sure both your comments are right – quadratic speedup for finding hash collisions, and the asymmetric crypto stuff is broken.

A.Sallai permalink

This is an incredibly well written article and one that i needed so much. Thank you, Michael!

Oli Rhys permalink

Thanks for this, while I understood the majority of it, the coding element was very useful – especially highlighting where the script goes in conjunction with the transaction.

While a lot of people know abot bitcoin, there is such a shortage of good quality technical info.

Thanks again 😉

Andrew Jaeger permalink

Great writeup about how bitcoin functions on a technical level, but I had a question about it as it's use as a currency.

Why is bitcoin built to be inherently deflationary? This seems to be the go-to argument against why it will ever gain widespread adoption as a currency. Why does the reward for mining bitcoin halve every 210,000 blocks? Could there be a point in the future where this is reversed?

Michael Nielsen permalink

Good questions, and I don't know. I certainly suspect (as do you) that these may ultimately turn out to be design flaws.

Assaf permalink

Bitcoin is NOT deflationary. It is inflationary with a known and decreasing rate up until around 2140 at which point it will stop being inflationary. The only deflation in Bitcoin may happen through coin loss. The same, by the way, is true for Fiat. The difference is that Fiat can be arbitrarily inflated and with Bitcoin it is not arbitrary.
As far as why inflation is predetermined, this is so Bitcoin is a better store of value which is one of the defining properties of a good "money".
Why is it inflationary at all (as in, why not start with a predetermined amount of bitcoins that never change). The is part of Bitcoin's decentralized design. Bitcoin designers wanted a way to spread bitcoins around without starting with a central authority that has them all and gives them out (like, say, ripple). The bitcoin generating part of mining does exactly that.

BTW/ I am yet to see a good argument about why having a monetary system that is a good store of value and does not get diluted over time with inflation is bad.

KRG permalink

Inflationary/Deflationary are properties of currency production relatively to the supply of real goods. Bitcoin is only not deflationary if you assume that real wealth production will gradually slow, and eventually stabilize around 2140 at the same pace as the drop in Bitcoin production. (And that's not even accounting for the effective tax pressure of the transaction fee needing to increase to attract the necessary computational power as the system of directly paying for that service phases out. The more that needs to be paid out in each transaction to cover the fees, the lower prices and actual payments will have to fall to make room for that overhead. Lower revenue translates to lower ability to afford a given price level, and so on.

Bitcoin has already seen hyperdeflation that rewarded early speculators, but the defined limitations in supply are far more likely to encourage more speculation over it as a commodity than less, making it's reliability for any predictable future value (never mind usefulness as an accounting measure to reliably store value) rather questionable.

What actually needs to be demonstrated is that there is any value in allowing any static, nonproductive account to maintain its nominal value, as opposed to using the inherent decline in the value of such accounts provide the baseline motivation to use more productive investments to store anything beyond cash sufficient to meet immediate needs for liquidity. There's no justification to use the monetary system to store value, because value is a property or real assets, not the money that serves to account for them. Trying to store value in money rather than in future production potential is the ultimate perverse incentive, rewarding fraud and financial manipulation far out of proportion to development of real assets.

mellyra permalink

Assaf is talking about inflation/deflation of the money supply, you are talking about price inflation/deflation.

Both usages are common in economics.

Jonathan Goldberg permalink

There are excellent reasons for wanting to store value. One obvious one is the desire to save for retirement.

Back in 1958 Paul Samuelson wrote a classic paper on the utility of money as a store of value:

An Exact Consumption-Loan Model of Interest with or without the social contrivance of money.
JPE V66 6 (Dec., 1958), 467-482

Dave **permalink**

Actually bitcoin is inherently deflationary if you believe that the size of the bitcoin economy will grow faster than the money supply. Although not quite intuitive, it does make sense upon reflection that the money supply reflects the value of the economy it represents. If the money supply is growing faster than the underlying economy then you get inflation. If the money supply is growing slower than the economy you get deflation.

I think all but a few of us expect the bitcoin economy to grow faster than the supply of bitcoins — hence we have a deflationary currency.

The wisdom of that choice is another mater, of course. One could imagine many different scenarios for the amount and timing (and conditions) of new currency entering the system. If bitcoin doesn't take-over-the-world then my bet is that these will be significant dimensions of experimentation among variants.

Wanton **permalink**

I'm confused about the block chain. Does everyone have their own version of it or do they sync to a master? Does every block chain get updated when validation is completed? Does this mean every person has a record of everyone else's transactions for ever? Won't this file get really really big?

Michael Nielsen **permalink**

In practice, there are thin clients which don't keep a full copy of the block chain. But the way the protocol is designed at present there is a sizeable number of people keeping a full copy of the block chain. This is currently quite a manageable size (about 12 gig). If Bitcoin grows rapidly enough this may eventually become a problem. There's a nice discussion of this and related scalability issues here:

https://en.bitcoin.it/wiki/Scalability

The conclusion there, which seems to me believable, is that there are many options for scaling Bitcoin at least up to the level at which credit cards are used today, and perhaps further.

Sunny **permalink**

Just about the total amount of bitcoins, if I understand well, new bitcoins are generated each time a transaction is processed? It means the more exchange we have, the more bitcoins in the market there is ?
So the only way to raise the number of bitcoins is to spend some energy validating transaction (that's a little bit wired for me ;-). How were created the first bitcoins ? Is there another way of creating bitcoins that checking transactions ?
And thanks a lot for this post because it's really difficult to get a clear picture of what is it. Regards

Marco **permalink**

1. Not per transaction but per block (of transactions).
2. Exchanges are a bad example. The transactions within the exchange happen outside the network. Only if you deposit or withdrawal BTC to/from an exchange, it goes over the network and therefore show up in the block chain. There are so many trades going on within an exchange, it happens internally. And since trades need to happen fast, the network is not suited for that.
3. yes
4. Google for the 'Genesis Block'. That's how it got started.
5. no

Daniel R. Grayson permalink

Very nice!

Comments: you use the concept of mining before defining it. Change “possibility vulnerabilities” to “possible vulnerabilities”. Fix “spending spending money”.

Michael Nielsen permalink

Thanks, typos fixed!

Greg permalink

Excellent article!

I found two typo’s which you might want to zap:

“Bob doesn’t just go ahead and accept the transaction. Instead, he broadcast Alice’s message to the entire network.” (Change ‘broadcast’ to ‘broadcasts’).

“Will Infocoin mining end up in concentrated in the hands of a few, or many?” (Remove first ‘in’).

Michael Nielsen permalink

Thanks, typos fixed!

Mango Cat permalink

Thanks for the great Bitcoin writeup. I have gleaned most of what you said in bits and pieces from articles and message boards, it’s nice to see it all described so clearly in one place.

What I think is more interesting than the cryptography aspect is the social-motivational aspect of Bitcoin and why it seems to be succeeding. First big mover and branding seem to be in-play, and the “anonymous cash” myth also was a big factor, but beyond that, I think the carrot of “get paid for solving hard problems (often using other people’s computing resources)” has drawn in many participants who have helped grow the network by promoting their own self interests.

I balk every time I hear the bit about “every transaction for all users for all time is encoded into the block chain” especially when combined with “the chain is developed by solving hard problems”. Scaling this system to support a billion users transacting multiple times per day seems…. unlikely. Your explanation does help to show how the problems don’t get much harder as transactions scale up – the blocks themselves get larger, but the hash problem doesn’t get significantly harder as block size grows, unless you start talking about transacting the world’s monetary business in such a system, then those blocks would get uncomfortably large in a very short time period, and the forking problem would be much more complex than choosing between two or three chains to follow.

I also somewhat disapprove of the concept of encouraging people to “mine digital space to earn currency” since that creates an artificial demand for energy which could grow into a significant waste of “real world” resources as such a system scales up.

I have been playing “trust network” thought games since the 1980s, I’d like to see a peer-to-peer digital currency system that is based in the concept of trustworthy digital identities instead of solving hash problems to get paid. You could still get paid by validating transactions, but it wouldn’t have that appeal of “solve more problems, get more paid…” plus, a system based on trust would tend to concentrate trust in central authorities that would be quickly perceived as hopeless to surpass in trustworthiness, semi-defeating the incentive to participate as a trusted member of the network, unless some kind of carrot to the underdogs was included – which would be purely social-motivational instead of a technically required component.

Anyway, all very interesting to watch. As usual, I got in late and out early with Bitcoin (bought around 5, sold around 120, seemed like an awesome profit margin at the time…) that aspect of Bitcoin is a lot like any other speculative investment, and is certainly fueling interest at this stage.

Michael Nielsen **permalink**

On scalability, check out https://en.bitcoin.it/wiki/Scalability. There's a lot of useful information there. Like you, though, I wonder about the long-run economics (and impact) of mining.

Michael Newman **permalink**

Thanks for writing this great explanation of Bitcoin. I noticed in the first Bitcoin transaction example, you mention 0.39 bitcoins, but the example really deals with 0.32 bitcoins, where 0.319 bitcoins goes to one person, and there is a 0.001 bitcoin transaction fee. In other words, did you mean "0.319" instead of "0.39"? Also, is there a need to show "0.31900000" value as an image?

Michael Nielsen **permalink**

Thanks, fixed.

Joseph Reagle **permalink**

typo: "trying to understand possibility vulnerabilities": possible

Michael Nielsen **permalink**

Thanks, fixed.

Jack L **permalink**

Thanks for the excellent writeup. I have a question about one item, hopefully you can explain it.

It appears the money you send someone is merely chunks of one or more previous transactions. Let's say I receive 1 bitcoin at myaddress_123 and I receive 1 bitcoin at myaddress_456. I now want to send you 2 bitcoins from myaddress_789. Those previous transactions are the inputs for my transaction to you.

How does the transaction message for the 2 bitcoin transaction prove that I was the recipient of those previous transactions when the addresses are all different? Is the hash for each input in the new transaction something that can only be generated by whoever was the recipient in the original transactions (myaddress_123 or myaddress_456)?

If the answer is yes, then it seems like unique addresses can be easily linked, in which case I don't see any anonymity advantage of using new addresses for each transaction. Sorry if I'm missing something obvious here.

Michael Nielsen **permalink**

The proof is in the digital signature. That signature is generated using a public key which must match (when hashed) the address from the output to the earlier transaction. That proves that the bitcoins are the payer's to spend.

ajay **permalink**

But (if I understand correctly) the need for every transaction to be publicly verified means that you are tied to all your transactions. You can't maintain a double life.

If I were a criminal, I might find it very desirable to have two personas – Stringer, who sells drugs, and Russell Bell, well-known property developer and pillar of the community – and use the cash that Stringer collects to bankroll Russell's legitimate businesses. But there's no way to do that with Bitcoin; I can't transfer Stringer's bitcoins to Russell without everyone else in the world knowing about it. Anyone with a copy of the block chain can notice that the flow of money goes from various drug users, to Stringer, to Russell.

Jim Lyon **permalink**

If you really want to enable money laundering, first create a bank. A bank would let any customer use bitcoin transactions to deposit and/or withdraw value, and the bank would keep its own record of individual customers' accounts. When withdrawing bitcoins, you would get bitcoins that were the bank's to give, but that were unrelated to the ones you originally deposited.

Such a bank would have more uses than just money laundering. It could pay interest, make loans (charging interest), convert bitcoins to/from more traditional currencies, etc.

Space2001 **permalink**

Extending from J. Lyon's answer.. Say you need to send $$ from one or more of your bitcoin addresses to a bad-guy but wish to anonymize this transaction. You will use a trusted middleman that does several transactions each day, some with good-guys and some with bad-guys. In one simple scenario the middle-man sets up a recievables address 'xxx' into which all the depositors send their bitcoins (i.e., the depositors all use xxx address as their output). The middle-man then transfers out the necessary amounts to intermediate addresses yyy0 … yyyM that he has set up specifically for this transaction period. After this is done, the middle-man provides the yyy* addresses one to each reciever. Because all the incoming money has gone into the xxx address there is no way to separate out subsequently which money went to which reciever. If ALL the yyyy addresses belong to bad guys then you would be guilty by association. Many bitcoin services perform such mixing by default, based on what I have read. The legal ramifications for the mixing service provider are unclear to me.

ajay **permalink**

Got it, thanks. But such a bank would have to keep its own records – botl as a practical necessity and as a legal requirement – and those could be obtained by the authorities. Whereas cash can be laundered tracelessly, through a cash business like a casino or restaurant, which can perfectly innocently be expected to have lots of cash coming in and no way of knowing where it comes from.

avee **permalink**

Interestingly this is exactly what was done with silk road. It basically was bitcoin bank moving bitcoins around in such a way the buyer and seller could not be connected. There's a paper which has some details about that: http://i.cdn.turner.com/money/2013/images/11/25/silk-road-paper.pd

Con Kolivas **permalink**

Nonce starting at zero is not a vulnerability. Shares are stochastically distributed throughout the 2^32 nonce range and it makes no difference where you start. The nonce is simply 32 bits out of the whole 320 bit coinbase that you are hashing and there is no way to design a target solution to be distributed anywhere within the nonce range of those 32 bits. If you start at a higher nonce value you simply will have less possible chances at finding a solution before you'll need to get/create a new coinbase to hash.

Alex F **permalink**

I think maybe you're misinterpreting his concern: the danger isn't that someone can solve blocks *strictly* faster by starting at a different nonce, because as you say, the correct nonce could be anywhere in the range 0..2^32 so every guess has the same 1/2^32 chance of being correct.

Instead, the danger is that someone could solve blocks faster *than everyone else* if they start at a higher nonce and everyone else starts at 0. Specifically, assuming (on average) everyone can calculate guesses at the same rate, then any transaction whose correct nonce is higher than X will always be solved first by someone who started at Y, if 0<Y<=X.

Of course this creates an obvious incentive for all participants to try to guess nonces in a different order than everyone else. So it seems reasonable that most client software would use a random sequence of nonce guesses rather than guessing sequentially from 0. But still, if one were to find a vulnerability in the random number generator of a popular client, then it might be possible to design a competing client which would, in practice, almost always find the correct nonce before the targeted client, by virtue of guessing the same sequence a few steps ahead. That would allow the attacker to successfully validate a share of blocks greater than their actual portion of the collective computational power, at the cost of everyone using the vulnerable client and finding the nonce less often than they should on average.

Gergely Imreh **permalink**

I think there's also a "time" field in the part that is hashed, which is also updated every few seconds. Thus in practice it's not the case that everyone has the same message and they just run the nonces – but everyone has a different message, regardless of the nonces. See https://en.bitcoin.it/wiki/Block_hashing_algorithm

Michael Nielsen **permalink**

Alex has explained my concern well. However, as a number of people have pointed out (including Gergely) in fact there are in any case small differences in the blocks being hashed by different miners, and that's sufficient to make this a non-issue.

Rubberman **permalink**

A most excellent and well written article! I look forward to more! Thanks.

Bram Stolk **permalink**

As people make transactions, the public ledger grows.
Will it not grow to an unmanageable size at some time?

Julian **permalink**

I have a couple questions, possibly a subject for a future article.

1. If the block chain forks, do the miners on both sides of the fork keep their rewards? If so, doesn't it allow someone to continue executing the proof of work even when it is known that someone else has solved the proof of work?

2. I am puzzled by transactions in blocks. Is it not possible for two miners to be working on different blocks which contain mostly, although not all, the same transactions? Then, the first one to solve the proof of work will have validated some of the transactions in the second miner's block. Does the second miner restart by taking his unverified transactions and putting them in a new block?

Michael Nielsen **permalink**

On 2, yes: if you're mining and someone else validates some of the transactions you are working on, then you remove them from your queue, but continue working with the unvalidated transactions.

On 1, it's true that in different forks, different miners will have been rewarded. However, over time only one of the forks will become the accepted consensus for confirmed transactions. And so only the miners from one fork will be able to redeem their transactions.

Rob permalink

Fascinating read, thank you!

One thing I'm having difficulty with is block chain integrity. What will happen when an owner loses his wallet restores a backup from a few weeks back. He may have spent some coins, and he may have received some. Those transactions are no longer in his block chain. How would the block chain get back in sync?

On your question-to-yourself about using two phase commit, I think the major issue would be vulnerability to denial-of-service attack. A malicious user could set up a swarm of identities to act as nay-sayers and therewith deny some or all others from performing transactions.

Benjamin Marty permalink

In my experience using the bitcoin client, you are not allowed to do anything on the bitcoin network until your block chain is in sync with the latest transactions. It somehow recognizes how far behind your block chain is and starts downloading blocks and tells you how old your block chain is and how much left you have to update as it downloads more. BTW, I un-installed the bitcoin client because over the 1 year span that I had it installed, the block chain went from about 2 GB to about 25 GB, and the novelty of having my own copy of the block chain wore off in comparison to its cost. It would be nice if there were some kind of "reset block" that could be generated that flattened the tree into a single block enumerating the value stored at each address.

Michael Nielsen permalink

On the naysayer DDoS attack on two-phase commit: if someone claims that they see evidence of double spending, you'd require them to present evidence in the form of a signed transaction. The requirement of a signature makes this hard to forge by a malicious naysayer.

uez permalink

Here is a very entertaining rational explanation
http://www.bringhurst.org/2013/04/03/how-does-bitcoin-crypto-work.html

Benjamin Marty permalink

One thing I still don't fully understand is how the bitcoin reward size is decided and awarded. Who enforces the rules that 25 bitcoins are awarded for validating a block, and a few years hence, it'll be 12.5 bitcoins? If we were to decide that the rewards should be different (remaining at 25 indefinitely, for example), what exactly would have to change? Is it the bitcoin mining clients that are hardwired to only validate transactions that award 25 coins to other miners when they validate their blocks, and the date of the validated block indicates that the award should be 25 BTC?

Michael Nielsen permalink

It's hardcoded, based on the number of blocks in the blockchain. Every 210,000 blocks the rate halves. No need to keep track of the date, simply count blocks.

Michal permalink

Hi,

And cannot miners just continue to validating the the transaction with adding 25 bitcoins?

As the chain is just validated list of transactions, how there can be any cap on transactions?
What does hardcoded mean practically?

Jozef permalink

You only own that much of bitcoins as others agree you own. So, hardcoded here means it is the original protocol suggested (and supposed) to be honored by all the users. There's no use for miner to continue giving himself 25 BTC when it's already time for lowering the reward to 12.5, because this reward would not be considered valid by others who respect the original protocol.

Anton D permalink

Would it be, in principle, possible for all miners to agree on not lowering the reward at all? For example to continue to reward 25 per block for all eternity.

Nick P permalink

I hadn't had time to thoroughly delve into the protocol and your excellent writeup is the exact piece by piece, what/why I needed. The "why's" are extremely important to people who might want to build on top of the protocol as it helps them understand what they should or shouldn't modify.

pebird permalink

I was thinking about how the blockchain is managed as more transactions are processed, thanks for the link https://en.bitcoin.it/wiki/Scalability

Interesting, one of the potential solutions discussed is the use of dedicated servers instead of lightweight clients to increase transaction rates, reduce latency, handle increasing blockchain size (via techniques such as "pruning" the chain), etc.

What this implies of course is an evolution into "banks", a group of entities with sufficient resources and staying power to dedicate specialized BTC infrastructure for transaction handling.

In a way, Bitcoin is replicating a history of money evolution in an accelerated manner. I wonder what will take place in the protocol to allow the peer-to-peer nature to continue while scaling the project to allow the transaction capacity necessary for a true currency.

Michael Nielsen permalink

Yeah, that is very interesting. I don't run a full client myself, I use a thin client that doesn't have a full copy of the block chain. And you do already see a lot of signs of centralization with the big mining pools:

https://blockchain.info/pools

Jeremy permalink

Great article. Thanks very much for writing it.

Reader permalink

Typo: requring = requiring

[MN: Thanks, fixed.]

Reader permalink

The concept of 'block' (and/or a definition) is not introduced before it is used. This makes the concept difficult to grasp.

Reader permalink

Wouldn't three phase commit be more appropriate than two phase commits ?

Michael Nielsen permalink

That would likely be even better, although I haven't thought about it in a lot of detail.

Reader permalink

Grammar:

to take account of those asymmetries => to take into account these asymmetries.

[MN: Not sure that was ungrammatical, but in any case I've improved it: "to account for those asymmetries."]

Reader permalink

Multiple typos with '…the recipient is receiving 0.39 Bitcoins…' -> 0.319

[MN: As noted earlier, fixed.]

Stuart Quimby permalink

Thanks for such a generous and informative post. There is so much babble on Bitcoin that it often seems to operate socially as more of a rorschach test on currency than an actual means of exchange. I quite agree that the details are considerably more interesting than yet another pundit's babble about what it all means. The devil, and the delight, are in the details.

Stuart Quimby

Terikan permalink

Bitcoin has fascinated me recently. I admit to not being able to fully wrap my head around it, but I took what I could and wrote a little here: http://mimictrading.com/viewtopic.php?f=4&t=293

Maybe you can help me out with one part of this I don't quite get. The signature. How does the block chain know that the address sending the coins is correct? The sender sends their sig to go with it, I assume paired up with the hash of the address allows the various nodes to validate right?

But if you are sending your sig out then can't any node have access to that private info. They would need to in order to validate. So can a sig only be used once, and if so how is it generated and what prevents it from being faked?

cryptograffre permalink

https://en.wikipedia.org/wiki/Digital_signature

Public key cryptography is a remarkable and beautiful thing. Each client using Bitcoin has keypairs – one key in each pair is public, the other private. The nature of asymmetric cryptographic digital signatures is that I can sign any piece of data using my private key, and anyone else with only my public key can verify that the person who signed that data holds the private key. There's some fascinating mathematics involved, exploiting a simple numerical relationship between the public and private keys.

richnormand permalink

Very nicely done write-up.

Makes me wonder about the news at various times about a major "theft" of bitcoins, mostly in exchanges. In order to benefit they would have to be converted or be re-introduced later on.

Some of these were for large amounts and not really easy to hide, unless you just "sit" on them?

Michael Nielsen permalink

I've wondered the same thing. Some observations: if you copy someone's private key, and then erase their copy, there is no way for them to prove that it was ever truly their key. And if two people both have a copy of the private key, how do you determine who "truly" owns it?

The situation is complicated further by the possibility of laundering. If you quickly spend some stolen bitcoins on, then it becomes very different to later recover those bitcoins, since now they may be in possession of honest parties.

---

shyam jos **permalink**

the best explanation ever , thank you Michael 🙂

Austin **permalink**

This was a fantastic article and answered all my questions about bitcoins.

Thanks!

---

someone **permalink**

What about the actual code? How many miners are using same piece of software?

Reader **permalink**

Indeed, this is a critical question.
The more implementations there is, the stronger Bitcoin would be, as it would not be dependent on the "features"or flaws of one particular implementation.
The apparent lack of unambiguous protocol documentation makes me think that alternative implementations are difficult to achieve.

Certainly, it would greatly help if there was some form of "RFC", or "ISO Standard", or "W3C spec" for Bitcoin.

Besmir **permalink**

HI,
Your article was very interesting and detailed, so I learned a lot more from BTC. I have one question or doubt: What is done with all these hashes? are they gonna be used for cracking/decrypting encoded data?
what is the real benefit behind generating hash tables?

thank you

---

muthuveerappan **permalink**

Did you do this video or is this video inspired by this post !! ?
http://youtu.be/Lx9zgZCMqXE – this too is good…

Michael Nielsen **permalink**

I didn't make that. I just watched a few minutes – it looks pretty good, certainly much more detailed and accurate than most of what's out there!

Michael Nielsen **permalink**

Many people have asked about scalability, so let me just leave this here:
https://en.bitcoin.it/wiki/Scalability

It doesn't address every possible concern, but I think the upshot is that there's a lot of room for Bitcoin to grow.

Haroun Kola **permalink**

Thanks. There's so much to learn about this currency and I'm loving all the attention that its getting.

Diogo permalink

Great article! I have a question: Could miners run a modified version of the software to choose not to publish a transaction in the blockchain? I mean, like a small group of powerful miners controlling the entire network?

Mark permalink

If you control half or more of the total mining power in the network, you can keep a transaction out of the blockchain by solving blocks faster (on average) than the miners who are trying to include that transaction. If you control less than half, you can delay the transaction, but sooner or later the rest of the miners will get ahead of you and your version of the blockchain will lose out.

jshell permalink

Great explanation — but doesn't solve this problem:

Bitcoins aren't actually backed by anything other than server time.

There was a time in this country when you can go to the bank and trade in your 20 dollar bill for an oz of gold. You can't do that anymore, b/c today dollars are back by debt not gold.

But bitcoins are backed by server time. That almost makes less sense than backed by debt.

Thomas Mahoney permalink

I think you're confusing an investment with a medium of exchange. An investment should be "backed up" by something, in the sense that it should give the holder a claim future cash flows or other real assets. But a medium of exchange is just that, something used to facilitate trade, an accounting device. It should have scarcity value and be resistant to counterfeiting. Fiat currencies have scarcity value to the extent that they are usually printed in finite amounts. Gold is generally scarce. And bitcoin is scarce as well.

Gold has been used as a medium of exchange for centuries. What is it "backed up" by? Nothing. It's just scarce, and therefore suitable as a medium of exchange.

Miksa permalink

BBC had an interesting article called "Why do we value gold?" few days ago.
http://www.bbc.co.uk/news/magazine-25255957

Marco permalink

If people are willing to pay for something that is rare or unique, it has a value.
A Ford Mustang '65, first issue Marvel Comic books, baseball cards, Gold, Bitcoins all have value because they're scarce and people are willing to pay for them. The demand for it defines the price.

Marc permalink

Excellent write-up, and I look forward to further installments – which leads me to ask: are you no longer updating your RSS feed(s)? I came here from Bruce Schneier's blog, and I like what I see so I subscribed (in goread.io), but neither of your feeds has anything newer than the beginning of 2013.

I'd like to clarify: I'm grateful for your posts, and I'm not complaining if you've dumped the whole RSS thing (Google did, why shouldn't you?). But if you _haven't_ dumped RSS, but it was supposed to be getting updated automatically… it isn't.

Michael Nielsen **permalink**

Thanks for pointing this out.

I just checked both RSS feeds, and they seem to be fine. I typically post longer essays, often in the 3,000-20,000 word range, which is why I only update my blogs a few times a year. You may enjoy looking through some of my past articles. This blog carries my more technical stuff, while my other blog (https://michaelnielsen.org/blog) is more general.

Your comment did make me notice and fix some mistaken links in my sidebar, so thanks for that!

Marc **permalink**

Huh. I clicked through to the Feedburner page, and indeed the new stuff is there. Perhaps the problem is on the go read side? I've been happy with it so far, but… Maybe later I'll try again with a different reader.

Here's what I see for the DDI feed: http://i.imgur.com/6T3TLN7.png
and here's what I see for your main feed: http://i.imgur.com/76XV1Ks.png

I've tried refreshing multiple times, but no joy.

I've bookmarked your pages and I can certainly come back here periodically to see if you've got anything new, but I thought you might like to know that (at least for some of us) RSS ain't Sing.

Michael Nielsen **permalink**

I checked in my RSS reader ("The Old Reader"), and both blogs seem to come through fine. Odd.

Marc **permalink**

It JUST NOW showed up in my feed (17 December, 21:00 Pacific time).
Where it's been all this time, I may never know.
Just thought I'd let you know.

Joseph **permalink**

Am I to understand that it takes about 60 minutes to pay somebody through the Bitcoin network? I reached this conclusion based on the 10 minute average block confirmation and the requirement of it being 6 back in the chain before it is considered confirmed.

Thanks,
-Joseph

Michael Nielsen **permalink**

Full confirmation requires about 60 minutes. Many people are willing to accept payment on more trust, though, say after just a single confirmation (~10 mins).

Yes, this is a significant disadvantage of the protocol in its present architecture.

Marco **permalink**

Transactions are instant. Confirmations are not.
A confirmation takes 10 minutes. If you want full confirmation, then yes, on average it takes an hour (6 confirmations).
For eCommerce, this will probably work in most cases.

For retail, this can be an issue.
However, there are a few points:
– If you try to double spend, it doesn't mean you will succeed.
– It is not easy to double spend in front of the cash register (unless you have build some app and are all prepared)
– The merchant (I think BitPay is doing this) can listen on the network to see if there was a double spend attempt. Those are easily detected.
– Don't forget that a Credit Card payment can be charged back 6 months later. Just saying. 🙂

Gregory Johnson permalink

Love the article. It is the first article that I have been able to understand on this topic, and I have been reading a few on it. And a comment to style, I really appreciated the higher-principled discussion on the topic. I am so annoyed with the internet's constant barrage of get-rich-quick articles on this, or the excitement of the exchange rate. I have yet to read before now any intelligent comments to the social value, in particular your link to http://szabo.best.vwh.net/formalize.html was appreciated.

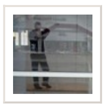
Michael Nielsen permalink

Thanks. All of Szabo's writing is worth reading, incidentally, his site is a treasure trove.

Steffen permalink

You know that there are some hints that Nick Szabo == Satoshi Nakamoto?

https://likeinamirror.wordpress.com/2013/12/01/satoshi-nakamoto-is-probably-nick-szabo/



Bart permalink

great article.

Ian permalink

Re: why BT doesn't use 2PC, as I understand it, it's because 2PC becomes exponentially more complex/unreliable with an increased number of parties.

2PC is a collapsed version of the byzantine / paxos protocols (which is 2PC with n in place of 2), and the basic problem is that a lot of nodes have to be online and swap a lot of messages in order for it to be workable. In any case it's more complex than 'longest blockchain wins'.

That's my amateur understanding anyway.

HPublius permalink

Great article and great discussion! This is a very good overview of the technical aspects around the bitcoin protocol. The fact remains that bitcoins have no intrinsic value and the promise of a peer-to-peer payment network (medium of exchange) will not be fulfilled unless the bitcoin is transformed into a true digital currency. Here are my thoughts on how to accomplish that: http://tinyurl.com/m57hd2z

OnlyMe permalink

Hi,
first of all great explanation on Bitcoin, I love it!
I guess my question is simple to answer.
How can I verify that a transaction is signed by a certain address if all I got is the hash of the public key? Don´t I need the full public key for that instead of only the hash?
What am I missing guys??

Michael Nielsen permalink

The transaction contains the Bitcoin address of the payee (or payees, if there are multiple outputs) in the output fields, and the public key(s) and signature(s) of the payer(s) in the input fields. So there's no problem — you do have the full public key of the payer.

kgb permalink

Silly question from a non technical person: how will transactions be approved and verified subsequent to 2140 when there are no more rewards for mining?

Michael Nielsen permalink

Transaction fees (which I briefly describe in a couple of places in the article).

Tom Hatcher permalink

Thank you. Best explanation I've seen so far! I still don't understand it completely, but it's slowly becoming clearer. One question, though. I hear that it's open source, and we can look at the source code. I'd like to do that. It's written in C++? Where can I find the code and look at it? Thank you.

Tom Hatcher permalink

I answer my own question. It's at github.com

Fee Fi Fo Fum permalink

I would love to see you discuss tumblers and the effectiveness and possibility of anonymizing your bitcoins

Cb permalink

In your anonymous section you speak of debunking a fairly huge myth without really backing it up. You just state the equivalent of "actually it's not anonymous" without going into detail.

I don't believe that to be true unless you are implying that various ways of using the internet anonymously are breakable.

For instance if TOR is compromised versus if it is not, or if other methods of obscuring traffic surrounding use of bitcoins are insufficient

Your assertion that bitcoin is open and transparent has nothing to do with its ability to use it anonymously, and the claim that it wouldn't be able to 'achieve convictions' but will narrow the pool of suspects down sound closer to a statement of successful anonymity rather than unsuccessful.

Michael Nielsen permalink

"You just state the equivalent of "actually it's not anonymous" without going into detail."

It's certainly not meant to be a proof! I do, however, go a great deal further than just saying "it's not anonymous" — I reference a large and growing body of academic literature that takes supposedly anonymous data sets and then de-anonymizes them. I believe techniques similar to those used in those papers will be very useful for attacking Bitcoin. There are complications in Bitcoin, notably that some people (though far from all) routinely use new addresses for each transaction. That makes an interesting challenge, and (I think) is different than in earlier work on de-anonymization. I'll be most curious to hear what the de-anonymizers have to say after making a sustained attempt at Bitcoin.

Magnus Sorenson permalink

Linking bitcoin addresses to a real identity requires that a real identity is somehow associated with an address in the first place. If I buy bitcoin on the street with cash – without revealing any personal information, using a public network, and a device with a mac address that can't be linked to me, then I can spend that bitcoin with absolute anonymity unless I reveal personal info to whoever I send that bitcoin to. In the case of Silk Road, the guy who sends me the drugs would need my mailing address, but that can be fudged as well. If he does not store my mailing address, nothing gets linked to me if the drugs arrive safely. Throw in a mixing service, and on that transaction, not even the NSA can ID me from the blockchain.
Now if i do the same thing many times, it may be possible to ID me using other vectors – but explain how anyone could ID me using the blockchain if I buy the bitcoin with cash on the street, and spend it leaving no permanent record. I assert that bitcoin can be 100% anonymous – forever- if you do it right.

MerkleTree permalink

The system is anonymous, but traceable. If you receive bit coins from somewhere and never spend them that's fine, but it becomes possible to track addresses back to wallets via "change addresses" that are created during man transactions.
There are several other methods as well.

I suggest you look up "BitIodine" . http://ow.ly/s2yZv

You will find that many Addresses can be identified with a bit of heuristic effort.

To remain anonymous, you have to take pretty extreme measures.
This includes the use of tumblers and foggers, but you cannot guarantee they will work.

This leads to the interesting point that if you steal bit coins, they are dangerous to spend.

Adam Back permalink

You mention using multiple sub-puzzles to reduce variance. This is a bad idea as it introduces progress. Because bitcoin is a first-past-the-post race, progress gives an unfair advantage to more powerful miners (>2x reward for 2x power).

Michael Nielsen permalink

You're presuming a particular design (in which people solve a set of puzzles privately). There is no necessity to make that presumption; this is why I make the first part of the statement "with some careful design it is possible to considerably reduce the variance in the time to validate a block of transactions". I haven't worked out the best possible design, however, I have found a design whose distribution is quite a bit more sharply peaked around 10 minutes than the current puzzle. Unfortunately, the details are more complex than I want to write out right now; I may come back to it in a future post.

Adam Back permalink

Also you talk about risk of nonce reuse. This wont happen because people mine for their own reward address, so even if the nonce is reused the work proof wont be.

Further in the case of pool mining the pools hand out work, specifically to avoid nonce re-use (which is somewhat insecure as others could guess the work range of other users and race them to produce it).

And finally the secure way is pooled miners use getblocktemplate and use a large random counter start extranonce. If extranonce is large enough and random the probability of nonce collision is pratically 0. You can read about this in the hashcash paper http://hashcash.org/papers/hashcash.pdf or https://en.bitcoin.it/wiki/Hashcash

For decentralization miners should also choose their own blocks by running as a full node and filling in the details into the coinbase provided by getblocktemplate.

Adam Back permalink

two-phase commit: if you are willing to wait 10 minutes, bitcoin already does that.

I presume the form it would take is the proof of double spend would be (one of) the double spends.

There have been proposals to forward double-spends with a double-spent marker (currently the first only is received).

Maybe just an api to ask if there are any transactions conflicting with a given transaction a user could ask a few random nodes to gain confidence.

You also have to bear in mind preserving the 0-confirmation spend functionality. Many people rely on that for low value point of sale transactions.

SRM permalink

"I'll return later to the question of why the Bitcoin address is a hash, and not just the public key."

I don't think you ever come back to this topic.

Michael Nielsen permalink

See the discussion above, in reply to Benoit Mason.

JohnT permalink

Thank you for the primer.

You might consider removing the footnote. IMO, Bitcoin cannot be successfully defended as free speech. Free speech is not a full blown unlimited right, as yelling fire in a crowded theater reminds us.

But Article I, Section 8, subparagraph ? does grant Congress full power to "coin money and declare [its] value." And a subsequent subparagraph grants Congress full power to outlaw any currency it wishes for citizen uses as legal tender.

Thus, IMO, the Supreme Court could never allow free speech to prevail over Congress's unfettered Constitutional authority.

I hope this comment does not derail a great discussion of Bitcoin. Please delete my comment if it becomes a red herring.

johnp permalink

a question
re the part about
“”
everyone (collectively) is the bank. In particular, we'll assume that everyone using Infocoin keeps a complete record of which infocoins belong to which person
“”
assuming millions and perhaps eventually billions are using bitcoins
so for example billions of transactions could take place daily
times this by the qty of bitcoins (each single one being unique)
times this by the billions using bitcoins
then what affect would that have on the network ?

Michael Nielsen permalink

Already addressed in comments above.

Pavel Masyuk permalink

That question about a nounce… I think that the parametres of the puzzle differs for every single miner. Everyone's desired block contains a unique transaction that no other miner has – a transation of giving a reward to himself. So there is no point in trying to trick others – parametres of their puzzles are different. Its OK for everyone to just try 0,1,2 etc…

Michael Nielsen **permalink**

Already addressed in comments above.

Joshua Holden **permalink**

So if I've got this right, one proof-of-work computation takes about 10 minutes, and you you (currently) get 25 Bitcoins for doing it, and each Bitcoin is (currently) worth about 1000 USD. Right? If so, this only makes sense if most proof-of-work computations don't get finished and/or don't get rewarded. Is that usually because someone else got there first? Do you know about what fraction of proof-of-work computations get rewarded?

This was really useful; thanks!

Steffen **permalink**

Joshua,

yes. It's a race. Whoever finds the hash that is smaller than the currently defined difficulty, they will gain the reward for the block. The difficulty is adapted every two weeks or so to reflect the changing (now growing) power of the network.
The power is growing so fast and so much that some already call it an "arms race".

-st

Chris Crozier **permalink**

Thanks for the article, very interesting, but I don't see any mechanism for re-combining fractional bitcoins.

I'm puzzled by what seems to be an ever-increasing fragmentation. It seems that over time you would accumulate a large number of coins of varying fractional values, and to make a payment you would have to lump together a collection of fractional coins to equal or exceed the transaction required, then typically end up with paying yourself your change. This one-way process of cutting off pieces of a bitcoin would continue steadily. A holding of one bitcoin would end up being constituted of maybe hundred or thousands of differently-sized fractions.

If that's the case, that will make transactions increasingly messy: you may have to consolidate a large number of inputs for one payment. In turn, that will lead to the block chain file growing faster and faster.

Did I miss something?

Michael Nielsen **permalink**

This is not a problem. Yes, some transactions will "fragment" bitcoins. But other transactions undo fragmentation. For example, a 5-input, 2-output transaction will reduce fragmentation.

This sounds a little complex for the user, but in practice, good client software will make this invisible. You simply say "I want to send such-and-such bitcoins to so-and-so address", and all the details of combining transactions will be taken care of. In this sense it's actually easier than cash, where we deal with the fragmentation / de-fragmentation problem all the time (i.e., finding the right combination of bills and coins to pay for a service, and then dealing with the resulting change).

Amit Prakash **permalink**

Amazing article. Looking forward for more on similar topics – may b someday you'll explain for tor.

Edward **permalink**

Very informative article; thank you. I was wondering:

1) With regard to transaction fees, I assume it is up to the first miner to successfully validate a block to determine if your offered fee is large enough to be included in the next iteration of the block chain. Additionally, I assume you have to determine what fee you're willing to pay well in advance of the next chain being verified. If this is the case and we fast-forward to 2140, won't all miners (assuming the computing power isn't concentrated) be incentivized to take ANY transaction fee no matter how small? For instance, if I have .001% of the computing power I should, on average, validate a block once every 2 years (hopefully the math's right, it would be infrequent in any regard) and, considering (assuming?) there would be no marginal cost to include a transaction (or 1,000 transactions) offering me only $.1 equivalent in transaction fees I can't think of why I wouldn't validate that transaction. As such, even if there were a few big players who controlled say 2/3 of the computing power, on average one in 3 blocks would be validated by a smaller player who wouldn't care about pricing; since people already wait nearly an hour to finalize a transaction likely they'd be willing to wait an extra ~30 minutes and, as such, the big miners would likely just lower their fee thresholds pretty substantially; Wouldn't this create an odd prisoners' dilemmaesque situation? Maybe I'm missing something but it strikes me that this would end with low fees forcing people out of business until computing power concentrated amongst an oligo(mono)poly of miners who could exercise sufficient pricing control which, as you pointed out, would probably create integrity concerns.

2) I believe I've previously read somewhere that there was a price threshold required to incentivize mining at different levels of complexity – for instance, if bitcoin were $1 tomorrow, would miners continue to mine or would the marginal cost of running the equipment outweigh the reward (ignoring fees)? If that's the case and a scenario like that occurs, does bitcoin grind to a halt or will some miners shut down or start running less expensive equipment as they did when bitcoin was in that price range? I guess this sort of boils down to whether the use of high cost computing equipment is a function of competition (and price) or problem complexity?

Both are hypothetical but I was curious to know if you (or anyone) had considered these questions.

Thanks again for the article.

Edward permalink

(I think) I figured out the answer to #2 – I was unaware of how difficulty was calculated.

JimmyWeg permalink

Great article! I'm working on a case and see that the bitcoin user employed more than a dozen different applications: Anoncoin, Phenixcoin, Primecoin, etc. I take it that the protocol is the same among the clients, though hash algorithms, proofs of concept, and the like may differ. From what I understand, if I use XPMs and want to buy something from a vendor who accepts BTCs, I have to go through some broker or exchange facility to complete the transaction. If that's correct, and considering the U.S. in a vacuum, isn't it like we're all carrying around a different brand of currency (USDs, Yen, Pounds, etc.) and have to exchange them almost every time we want to buy something? It's easy with credit cards, but I don't see a similar approach with bitcoins. Thanks!

David permalink

"I don't understand why double spending can't be prevented in a simpler manner using two-phase commit. Suppose Alice tries to double spend an infocoin with both Bob and Charlie. The idea is that Bob and Charlie would each broadcast their respective messages to the Infocoin network, along with a request: "Should I accept this?" They'd then wait some period – perhaps ten minutes – to hear any naysayers who could prove that Alice was trying to double spend. If no such nays are heard (and provided there are no signs of attempts to disrupt the network), they'd then accept the transaction. This protocol needs to be hardened against network attacks, but it seems to me to be the core of a good alternate idea. How well does this work? What drawbacks and advantages does it have compared to the full Bitcoin protocol?"

I think something along these lines is planned:

"On top of all that is a long list of new features and improvements I'd like to see get into a 0.9 release; the highest priorities on my wish list are:
1."First double-spend" relay and detection. Detecting attempted double-spends as soon as possible is great for low-value, in-person transactions, and we should do more to support that use case."

https://bitcoinfoundation.org/blog/?p=204

David permalink

Great article. I will use it as a self-study tutorial.

In your next instalment, could you give a broad description of where the protocol is actually to be found (is it a particular piece of software?), how can it be changed, who can change it and indeed to what extent is it capable of being changed? These are important questions because they go to the ability of Bitcoin to evolve and develop, but it is very hard to find any good general account of these issues.

Best regards

David

Nitan Shalon permalink

If Satoshi Nakamoto already made two patches to Bitcoin, what's stopping him/her from making another patch right now that destroys Bitcoin?

Arthur Colle permalink

"Of course, after Alice has published her message it's possible for other people to duplicate the message, so in that sense forgery is possible. "

How can someone forge the message without Alice's private cryptographic key?

Dave permalink

Hi. Great article. Thanks for putting it together.

I am still having one big problem — and I feel like I must be missing something obvious. You wrote:

"Suppose Alice tries to double spend with Bob and Charlie. One possible approach is for her to try to validate a block that includes both transactions. Assuming she has one percent of the computing power, she will occasionally get lucky and validate the block by solving the proof-of-work. Unfortunately for Alice, the double spending will be immediately spotted by other people in the Infocoin network and rejected, despite solving the proof-of-work problem. So that's not something we need to worry about."

Who is going to be looking to reject it, and what does that even mean? If a malicious party (Alice) manages to complete a block that contains transactions that are not, in fact, valid then what? Do other miners check them before building on top of her faulty block? And, if not, then what does it mean for others to 'spot' them.

This has been bugging me for days! Your thoughts would be greatly appreciated.

Michael Nielsen permalink

Anyone with a copy of the block chain is not going to accept an additional block which has an obvious attempt to double spend in it.

Dave permalink

Thanks!

So that means that miners examine each block for conflicts before they choose to build on it…? And, if that's true, how is the overhead of that inspection built into the incentive structure? Will their block (if they solve it) become invalidated if down the line someone points out that they built on a block with a double spend? (That would make sense, but it isn't a step that I have seen described.)

Michael Nielsen permalink

Checking for a double spend within a block is computationally negligible.

Daniel Park permalink

That's a long detailed process of bitcoin above which was a great read. I don't quite understand near the end, but I definitely get the gist of it.
I don't perfectly understand the relationship between transactions and blocks (is this only required when ppl try to cheat the system?) If so, the money earned by Miner essentially is imaginary and something that only exists within trust that bitcoin is going to continue to work. I guess there are two cases:
1. as miners get bitcoins, the value of bitcoin comparison to other currencies drop. (This is probably the case)
OR
2 as miners get bitcoins, if it comes to a point where everyone wants to convert their bitcoins to real currency, it's not going to equal each other; bitcoins -> conversion via bitcoin exchange rate -> not enough real money.

Also, why assume every 210,000 blocks is occurs every 4 years? is this an assumption based on bitcoin flow so far? Wouldn't every 210,000 blocks occur more often if there is more flow?

Steffen permalink

If everybody would like to exit Bitcoin at the same time the price would collapse. The current speculation is though, that the opposite is true. Many people try to buy bitcoins for the fiat money. How many? Please see here http://fiatleak.com.

With regards to why 210,000 blocks are created in roughly four years: The network difficulty is set so that only six blocks per hour can be created. Roughly every 10 minutes a new block enters the blockchain.

Let's do the math: 4 years ->1461 days * 24 hours -> 35,064 hours * 6 blocks -> 210,384

Jozef permalink

Fairly good explanation although some important things missing. One mistake though:

"So if we want the output hash value to begin with 10 zeroes, say, then David will need, on average, to try 16^10 (approx 10^12) different values for x before he finds a suitable nonce."

I think it should be 16^9 on average. At worst it is 16^10.

Michael Nielsen permalink

No, the average number of trials is 16^10.

(If we repeatedly sample from a Bernoulli random variable with probability p of a success, then the expected number of trials to success is \sum_{n=1}^\infty n p (1-p)^{n-1}, which is easily shown to be 1/p. The probability p in the proof-of-work is 16^{-10}, so the expected number of trials to success is 16^{10}.)

What important things are missing from the explanation?

Lex permalink

i have alot of questions about bitcoin and ive done alot of searching, this explanation has to be the one that has got me the closest to the answers ive been looking for, unfortunately im still falling short of understanding alot of the basic, i do want to ask 1 question here which seems to be the most important question for me. In computer science terms what is a "bitcoin" there were points in this article where it seemed close to specifying this, although i may have actually missed the answer, so, what is a "bitcoin"?? is it a unique value? if i have 100 bitcoins what is it that i really have ? 100 unique values ? or is it simply a value that was "said" to be giving to me and simply assigned a value, sort of like the 5th entry in a ledger, in the genesis block it was "said" 50 bitcoin was givin to whom ever,50 bitcoin was given to bill and we will call that 50 1234567.. ??? overall im still searching for clarity on the fundamental idea of a bitcoin from a computer science view

---

david lee permalink

Michael,
We are organising World Bitcoin Conference on 24 & 25 March 2014 in Malaysia. Would like to invite you to speak on Bitcoin protocol. Could we have your email address to send the invitation.

David permalink

Hi – what a great write-up! As for your first Author's question – There are probably several less complex methods for confirmation but there is inherent security in the current approach which appears to organically solve the problem without risky reliance on factors and layers outside the network itself.

---

momingle permalink

There is one part that I am not sure I understand. For example, let say miner A has on his queue transaction A, B, and C to validated on a new block. Is it possible miner B will have transaction B, C and D in his queue (but not A) that he will validate in his new block? Assuming both solve the puzzle. Now both transactions B and C are in two different blocks. Will both blocks got accepted?

Nagaraj Hubli permalink

Thanks for the write-up, it helped my a lot in understanding the underlying tech of the Bitcoin protocol. Can't wait for the next in series.

---

elvin permalink

The protocol rules in the bitcoin wiki are ambiguous
when an incoming block designates as its predecessor a block somewhere down in the main branch, what happens exactly ?
the wiki can be interpreted in two ways :
1. nothing at all – the incoming block is NOT added as the beginning of a side branch
2. the incoming block IS added as the possible beginning of a side branch, but without any verification for the moment. If that is the case, then most blocks in the main branch should have many "brothers"

---

elvin permalink

I have verified that the correct answer is 2.
I have also found what limits the number of "brothers" for a block
1. miners stop mining as soon as they receive a valid node and therefore no new blocks are sent
2. their neighbours do not relay the blocks they might have sent in the meantime
Sorry for asking

---

fivebuckchuck permalink

Thanks for the write up ..

How do transactions get organized into blocks ? Are nodes broadcasting transacations or blocks ?

Number of transactions in a block ; is that hardcoded in the protocol ?

in your example of h("hello world"|nonce) , is "hello world" a unique transaction or the whole block ?

Rich permalink

“Suppose Bitcoin mining software always explored nonces starting with x = 0, then x = 1, x = 2,\ldots. If this is done by all (or even just a substantial fraction) of Bitcoin miners then it creates a vulnerability. Namely, it’s possible for someone to improve their odds of solving the proof-of-work merely by starting with some other (much larger) nonce. More generally, it may be possible for attackers to exploit any systematic patterns in the way miners explore the space of nonces….”

This is incorrect:
Because the block hash is dependant on the contents of the block. For there to be any possibility of a miner improving his odds through this method the miner must be mining the exact same block as someone else including not using his own address for coinbase and transaction fees to go to. Removing the entire incentive for mining.

Trackbacks and Pingbacks

1. How the Bitcoin protocol actually works | Enjoying The Moment
2. Understanding the Bitcoin protocol | Minsheng 民生
3. Science & Technology Roundup: #3 | Great Wall of Numbers
4. mid-day-ish links dump | imagine art -||+ there about?mid-day-ish links dump
5. How the Bitcoin protocol actually works | DDI | geistwc
6. How the Bitcoin protocol actually works | DDI | Mark Solock Blog
7. How the Bitcoin protocol actually works « adafruit industries blog
8. The Bitcoin Protocol - iamronen
9. Links 12/8/13 « naked capitalism
10. Bitcoin for dummies – Author walks users through how Bitcoin actually works | My Blog
11. Want to Spend Bitcoin in France? (Caveat Emptor) | FrenchNewsOnline
12. Robert McGhee » December 8th
13. ¿Cómo funciona realmente el protocolo del Bitcoin? [ENG]
14. BitCoins
15. Bitcoinizzle For Shizzle | BubbleCoin
16. How the Bitcoin protocol actually works | Blog of Leonid Mamchenkov
17. Starting of “Gift Economy (reputation based)” | l'ambizione dell'artigiano
18. Headline News from PaymentsNews.com – December 9, 2013 - ResidualMarketplace.Com – Submit Residual Listings for Free
19. How the Bitcoin protocol actually works « The Bitcoin Channel
20. Bitcoin Explanation « Electronic Padlock
21. Bitcoin Explanation
22. Bitcoin | Thoughts on computing
23. Links: Mandela and Great Leaders; WTO deal in Bali; How Money Works
24. How Does Bitcoin Work? ← Clinton Pavlovic
25. Following Bitcoin Believers is still an act of faith | Digital Trends
26. Virtual Mining Bitcoin News » Bitcoin Believers aren’t wrong, but don’t mistake their faith for fact
27. Open thread: Bitcoin and similar currencies
28. How Bitcoin Actually Works
29. Bitcoin Speculations | Aleph
30. How the Bitcoin protocol actually works | The Freedom Watch
31. How the Bitcoin protocol actually works | DDI - The Ripe Group
32. Explaining Bitcoins | kef's blog
33. A Detailed Explanation of the Bitcoin Protocol – Glimpse From a Height
34. Friday Links | Meta Rabbit
35. Bitcoin
36. Virtual Mining Bitcoin News » Bitcoin: Open source money whose time has come
37. Bitcoins, Cash and Gold | GusBook
38. Plaidoyer pour le Bitcoin | ComplexitésComplexités
39. Occupy BitCoin
40. How to not get rich from bitcoin | Turing's Invisible Hand
41. Bitcoin: The Economic Frontier of a Digital Age | The American Statesmen
42. Why You Need to Own Bitcoin | Winning Bitcoin
43. Bitcoin: a bit risky | Learn About Finance Careers
44. Various well written articles/sites | GSR8 Blog
45. Bitcoin transactions explained; How the Commodore 128 was … | BitCoin Razor
46. Digitopoly | Time for a little Bitcoin discussion
47. How and why BitCoin will plummet in price
48. How the Bitcoin protocol actually works and what that means for us | Helping Public Markets Grow
49. | Explaining bitcoin in a bar
50. Cryptocurrency Cat-and-Mouse games in China | Great Wall of Numbers
51. Random thoughts: Bitcoin
52. Info On Bitcoin | Curly's Blog
53. My Favourite Bitcoin Bookmarks | Webs Wonder Design - Webs Wonder Design
54. Facecoin In More Detail | Rob Myers

55. Fed Banker Tries Criticizing Bitcoin, Ends Up Perfectly Describing The Fed – Bitcoin Magazine
56. Bitcoin 2.0: Distributed Corporations, Derivatives, and Information Markets | The Ümlaut
57. The Difference between Bitcoin and OpenTransact | OSCurrency
58. j.r.mchale : The Nuts-and-Bolts of the Bitcoin Protocol
59. Fed Banker Tries Criticizing Bitcoin, Ends Up Perfectly Describing The Fed
60. Sécurité Bitcoin #1 : Introduction | Le Comptoir Sécu
61. De week in 375 woorden « Bits of Freedom
62. Bitcoins the hard way: Using the raw Bitcoin protocol | Bitcoin
63. On Bitcoin | Emergent Chaos
64. bitcoin critical mass/explosion, wheres it going? | Turing Machine
65. Bitcoin Beyond Money | SoshiTech – Social Media Technology – Soshitech.com
66. Bitcoin reading for developers | Coffee Spoons of Code
67. The Homebrew Litecoin Mining Project | Record important events of digital currencies
68. Complete Dogecoin Guide - Introduction into Mining - Everything Doge!
69. How the Bitcoin protocol actually works | DDI | Arca
70. bitcoin | jamesmath
71. Fun economic reading | MuddyHorse Farm and Tech
72. Bitcoin: have some of us been had? | Jim Bennett
73. “The Bitcoin Protocol” Infographic |
74. I don’t get Bitcoin… but I want to… | StartupCafe
75. Daily BTC – PBOC edition : Token Intelligence
76. Enter The Blockchain: How Bitcoin Can Turn The Cloud Inside Out | TechCrunch
77. Quora
78. Bitcoin Family Tree | Randolog

Comments are closed.