How Bitcoin Works Under the Hood

new 3.5 in-depth course: Introduction to Bitcoin, Blockchain and Decentralized projects (Ethereum).

1. Intro

The goal of this video is to explain how Bitcoin works under the hood, to give a clearer idea of what it really means to own, send or “mine” Bitcoins.

If, instead of how it works, you're looking for where to buy Bitcoin, I use coinbase. And for trading, check out bitcoin wealth alliance (both affiliate links).

2. What is Bitcoin at a high level?

First, a brief high-level overview of what Bitcoin is.

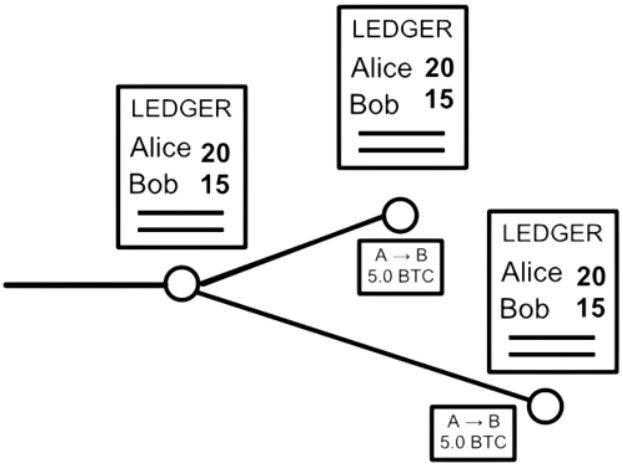
Ledger

Alice	5.3
Bob	100
Frank	700
Carlos	3
Jane	1.3
Charlie	4.645
Scott	.00000001
Kristin	1

...

At its core, Bitcoin is just a digital file that lists accounts and money like a ledger. A copy of this file is maintained on every computer in the Bitcoin network. (update: you don't have to maintain a ledger just to use Bitcoin to send and receive money, this is for people who want to help maintain the system).

These numbers don’t represent anything in the physical world, they only have value because people are willing to trade real goods and services for a higher number next to their account, and believe that others will do the same. The numbers only have value because we believe they have value, just like any other fiat currency.



To send money, you broadcast to the network that the amount on your account should go down, and the amount on a receiver’s account up. Nodes, or computers, in the Bitcoin network apply that transaction to their copy of the ledger, and then pass on the transaction to other nodes. This, with some math-based security, is really all there is--a system that lets a group of computers maintain a ledger.

While this may sound similar to the way a bank maintains a ledger, the fact that the ledger is maintained by a group rather than a single entity introduces a number of important differences. For one, unlike at a bank where you only know about your own transactions, in Bitcoin, everyone knows about everyone else’s transactions.

Also, while you can trust your bank, or can at least sue it if something goes

wrong, in Bitcoin, you're dealing with anonymous strangers, so you shouldn't trust anyone. The Bitcoin system is amazingly designed so that no trust is needed--special mathematical functions protect every aspect of the system.

The rest of this video will explain in detail how Bitcoin allows such a group of strangers to manage each other's financial transactions.

3. How Sending Money in Bitcoin Works

At a basic level, for Alice to send money to Bob, she simply broadcasts a message with the accounts and the amount: "Send 5.0 BTC from Alice to Bob."

Every node that receives it will update their copy of the ledger, and then pass along the transaction message. But how can the nodes be sure that the request is authentic, that only the rightful owner has sent the message?

Bitcoin rules require a kind of password to unlock and spend funds, and this password is what's called a "Digital Signature." Like a real handwritten signature, it proves the authenticity of a message, but it does so through a mathematical algorithm that prevents copying or forgery in the digital realm.

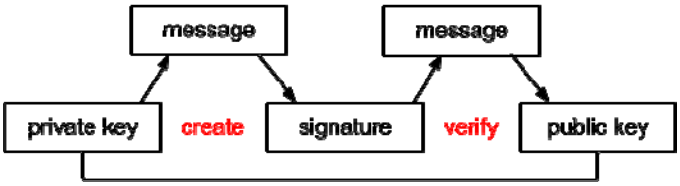
Unlike a simple static password, a completely different Digital Signature is required for every transaction. Keep in mind that in Bitcoin, you're dealing with complete strangers, so you never want to reveal a password that could be copied and reused by someone else.

Transaction Messages

		Digital Signature
Alice → Bob	5.0 BTC	04323784...
Alice → Dave	12 BTC	88432738...
Alice → Juan	2000 BTC	00328434...
Alice → Bob	14 BTC	19382637...

^
different every time

A Digital Signature works by utilizing two different (but connected) keys, a "private key" to create a signature, and a "public key" that others can use to check it.



You can think of the private key as the true password, and the signature as an intermediary that proves you have the password without requiring you to reveal it.

Public keys are actually the "send to" addresses in Bitcoin, so when you send someone money, you're really sending it to their public key.

1427L1ARMZ2AP2oHdUhwY9vuLCfGqfgX2u 50 BTC → 15ijJSSPMw9wkCnaUoXuwCxFLeXAtoW

To spend money, you must prove that you're the true owner of a public key address where money was sent, and you do that by generating a Digital Signature from a transaction message and your private key.

signature = f(message, private key)

Other nodes in the network can use that signature in a different function to verify that it corresponds with your public key.

1 =? v(message, public key, signature)

Through the math behind the Digital Signature, they are able to verify that the sender owned a private key without actually seeing it.

Importantly, because the signature depends on the message, it will be different for every transaction, and therefore can't be reused by someone for a different transaction. This dependence on the message also means that no one can modify the message while passing it along the network, as any changes to the message would invalidate the signature.

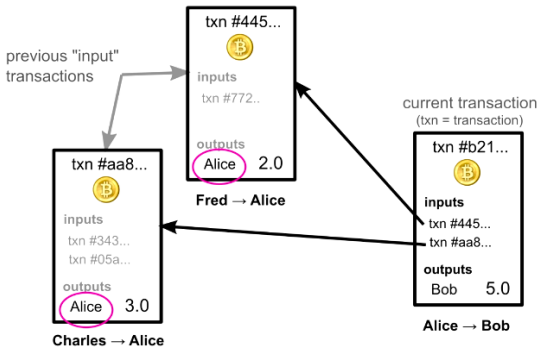
The math behind this is fairly complex, and while I won't try to explain it fully now, here are some topics you can google to get started: ECDSA and mathematical trap door. More at the end of the video.

4. Bitcoin Transactions and Ledger in Detail

So far, we know that Digital Signatures are used to ensure a transaction is authorized, but I've over-simplified how nodes in the network keep track of account balances. In fact, no records of account balances are kept at all. If you don't keep track of how much money any given person has, how do you know if they have enough to send to someone else?

Instead of balances, ownership of funds is verified through links to previous transactions. Here's how this works.

To send 5.0 BTC to Bob, Alice must reference other transactions where she received 5 or more Bitcoins. These referenced transactions are called "inputs." Other nodes verifying this transaction will check those inputs to make sure Alice was in fact the recipient, and also that the inputs add up to 5 or more Bitcoins.



Let's look at a real transaction to see this in practice.

Inputs

Previous output (index)	Amount	From address	Type	ScriptSig
eb38f77560ca...1	8	1P9SgqqjFWgWVAuZBFwimNPV7LunaJpgTj	Address	3045022007 8df7c48ed152bd40eaee4a73afefc3 044760639da2c0d6158484e1a4dab332fefc4bb
b912994fca58...1	0.03	18Mk65wV1E5kCVHFShvUTU6zt4yVFKM5Ft	Address	304502204e877fc5ca3783e165052e64c4788dd 04769bb8c55cbd412784e024c8624f8c4f42d7cb
58379d94fe85...15	1	1G4hfnM2nfAPEECdawg5gtvUTBB2PwvLr2	Address	3044022075d236d4a80048667772 10f51f46c96 046dd45b37fe3ff33f1563458cfbdfb7f922d1b4a
fc9d1cd1c2ac...1	130	1LpQVuJSMgqqhQBGZwbobsfN2Ghn9YWyC7	Address	3046022100a65a188b89a4e5ae2eaa5ba387503 04ba81a1a538c5ddf7e0c76884497ab522456b9
7b6f7d4a521c...1	0.55357267	16Kb6XppHUbjgnYQDpRvxs9jNE9Az5Xvcb	Address	3045022100eeb76e61abe62d38fd462eafd1d11f 04f4fa1d3e26f3e7058038871a31b8bf636fd127f6
544097a30e09...0	0.03270607	1JnsDx1g6c757s8AnJUemj46YQgCTw54QN	Address	3045022100859df22ced47493e86a849cce10615 04de25f7fe6490bd16188be6d06ca7b34816fa4b
Outputs				

139.6

Outputs

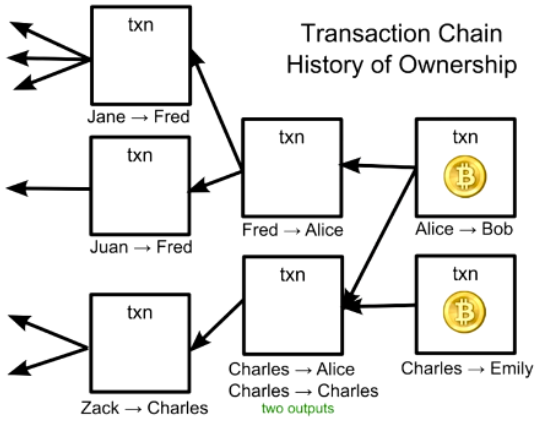
Index	Redeemed at input	Amount	To address	Type	ScriptPubKey
0	8baaca27d158...	0.01071174	1E7BgzQ8yWTWzEMUKNzLdjkbjaQT9K96m	Address	OP_DUP OP_HASH160 9abd2e0c0a63dea36b75c3128fe15d82f274e39 OP_EQUALVERIFY OP_CHECKSIG
1	1bb973b4ccc8...	139.605567	1NT2sFMa11NiCZvdt4kqpXRZPf3iS6ZPGZ	Address	OP_DUP OP_HASH160 eb471d7a903e538cb94c1f2faf20eaadad8479af OP_EQUALVERIFY OP_CHECKSIG

139.6

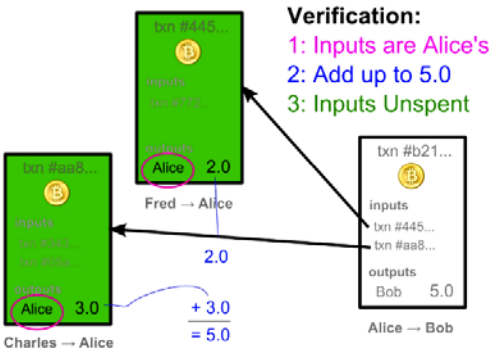
http://blockexplorer.com/tx/a117c441aa5bd3fcb442e3c47a180c584420bcd9f93c68dab9feddd1d26b767e

This transaction references 6 inputs for a total of 139.6 Bitcoins. In the output section, notice that there are two lines. The first one of these is actually going back to the sender as change for the transaction. A simplifying rule states

that each input must be used up completely in a transaction, so if you're trying to send an amount that doesn't exactly match one of your inputs, you need to send any remaining amount back to yourself.



Through these referenced input linkages, ownership of Bitcoins is passed along in a kind of chain, where the validity of each transaction is dependent on previous transactions. But how can you trust those previous transactions? You can't, and should check their inputs, too. In fact, when you first install Bitcoin wallet software, it downloads every transaction ever made, and checks each one's validity all the way back to the first transaction ever made. Remember, you're dealing with complete strangers, so it's important to verify every transaction yourself. This process can take over 24 hours, but only needs to be done once.



Once a transaction has been used once, it is considered spent, and cannot be used again. Otherwise, someone could double-spend an input by referencing it in multiple transactions. So, when verifying a transaction, in addition to the other checks, nodes also make sure the inputs haven't already been spent. To be explicit, for each input, nodes check every other transaction ever made to make sure that input hasn't already been used before. While this may seem time consuming, as there are now over 20 million transactions, it's made fast with an index of unspent transactions.

So, instead of a ledger of balances, Bitcoin nodes keep track of a giant list of transactions. Owning Bitcoins means that there are transactions in this list that point to your name, and haven't been spent, or, in other words, used as inputs in other transactions.

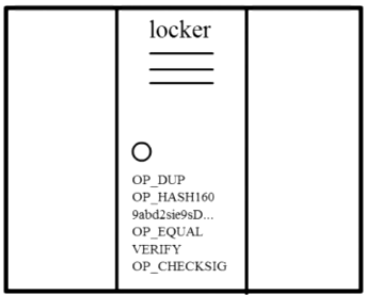
One interesting consequence of this ownership structure is that figuring out your own balance requires iterating through every transaction ever made and adding up all your unspent inputs.

Another interesting note about transactions is that the system can support more complex ones than simply sending funds to one person. You may have noticed a cryptic looking line of text in the output shown previously.

OP_DUP OP_HASH160 9abd2e0c0a63dea36b75c3128fe15d82f274e394 OP_EQUALVERIFY OP_CHECKSIG

http://blockexplorer.com/rawtx/a117c441aa5bd3fcb442e3c47a180c584420bcd9f93c68dab9feddd1d26b767e

It turns out that outputs are more like puzzles to be solved rather than simple “to addresses.” Rather than emailing, sending money in Bitcoin is more like putting money in a public locker and attaching a math puzzle that must be solved to open it. The puzzle is defined using a special scripting language, and while it’s typically designed so that only a single owner of a public key can solve it, more complex conditions are possible. For instance, 2 out of 3 signatures could be required for an escrow based transaction. Another example is the very first bitcoin transaction ever made, which was a puzzle that anyone could solve.



While most Wallet software hides this scripting layer for you, you’re free to write your own software and claim conditions, although this can be risky. Over 2600 BTC were lost in one batch of transactions due to a malformed address.

This highlights an important part of Bitcoin. As there is no bank or credit card company you can appeal to, any “user-error” mistakes can result in the permanent loss of Bitcoins, not just from your own account, but from the Bitcoin economy overall. If you lose your private key, any funds associated with the corresponding public key will be gone forever. Because people will likely lose private keys due to hard drive crashes and insufficient backups, this means the Bitcoin currency will eventually be a deflationary one.

5. Anonymity

Before explaining the final piece that secures Bitcoin (“mining”), I want to highlight a few points about anonymity in Bitcoin.

If you access Bitcoin through a TOR network that hides your IP address, you can use Bitcoin without ever revealing anything more than your public key. And to avoid someone linking your transactions together (remember, they’re all publicly stored on every computer!), you can generate a new public key for every incoming transaction.


Inputs

Previous output (index)	Amount	From address	Type	ScriptSig
eb3877560ca...1	8	1P9SgqqFWgWVAuZBFwmNPV7LmuaJpgTj	Address	3045022007 8df7c48ed152bd40eaee4a7 3a6efc30 0447606639da2c0d6158484e1a4dab332fdfc40b8c
b912994fca58...1	0.03	18Mk65wV1E5kCVHFSbvUTU6n6vYFKM5Fs	Address	304502204e877fc5ca3783e16505 2e64c4788dd 04769bbfc55cbd412764e024c8624f8c4f42d7c8
5837949946f5...15	1	1G4tbMJdAPEECdawgfgtvUTBB2PrvLr2	Address	3044022075d23fdd4a80048667772105f1f46c99 0466dd45b3f5e3ff33f1563458cfbdfb7f922d1b4a
6c9d1cd1c2ac...1	130	1LqQVuNSMgqqbQBGZwbobdX2Ghn6YWyC7	Address	3046022100a65a188b89a4e5ae2eaa5ba387503 04ba81a1af538c5dd7e0c76884497ab522456b9
7b6f7d4a521c...1	0.55357267	16Kh6XpgsHUbqmYQDpRxz9jNE9Ax5Xvcb	Address	3045022100eeb76e61abe62d3f8fd462ea6d1d11 04f6d1d3e26f5e705803887 1a31b8bf63fd12786
544097a30e09...0	0.03270607	13mDs1g6c757sfAnfUmqi65YQgCTw54QN	Address	3045022100859df2ced47493e86a894cce10615 04de25f6e6490bd16188be6d06ca7b34816fa4b

Outputs

Index	Redeemed at input	Amount	To address	Type	ScriptPubKey
0	8baaca27d158...	0.01071174	1F7BgtQbyWTWzEMUKNzzLdjkbzsQT9K96m	Address	OP_DUP OP_HASH160 9abd2e0c0a63dea36b75c31286e15d82f27e39 OP_EQUALVERIFY OP_CHECKSIG
1	1bb973b4ccc8...	139.605567	1NT2zFMa11NCZydr4kqpXRZPfiS6ZPGZ	Address	OP_DUP OP_HASH160 eb471d7a903ef38cb94c1f2fd20eaadad8479af OP_EQUALVERIFY OP_CHECKSIG

Bitcoin transaction that combines public keys

It is possible, however, to inadvertently link public keys together. In the transaction shown earlier, 6 “input” transactions were used as

sources, and despite the fact that all those inputs were sent to different addresses, they all became linked in that transaction. The sender proved that he owned all of the addresses by supplying the Digital Signature to unlock each one. Researches have, in fact, used these links to study Bitcoin user behavior. See Quantitative Analysis of the Full **Bitcoin** Transaction Graph by Dorit Ron and Adi Shamir.

You might think that generating a public key "receiving address" could potentially create a link to your true identity, but even this step is anonymous, and amazingly, can be done with no connection to the network. You simply click a button in your Wallet software, and it randomly generates a new private and public key. Because there are so many different possible addresses, there's no reason to even check if someone else already has that key (compare this to signing up for an email address, where almost everything you might try has been taken). In fact, if you did guess someone else's key, you would have access to their money!

This is the total number of possible Bitcoin addresses:

1461501637330902918203684832716283019655932542976 (1.46 x 10^48 or 2^160)

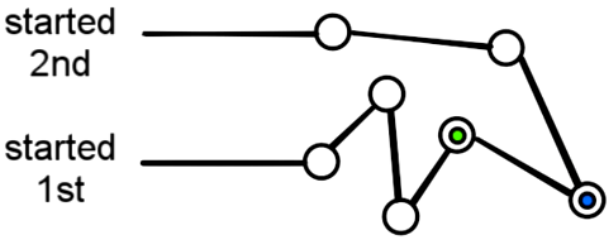
These large numbers protect the Bitcoin system in several ways, so it's useful to try to appreciate just how big they are. Some estimates for the number of grains of sand in the entire world are around 7.5 x 10^18th, or 7,500,000,000,000,000,000. Now imagine that every grain of sand represented an entire other Earth of additional grains, and you're still much smaller than the possible number of Bitcoin addresses.

(strictly speaking, the probability of two addresses matching gets reduced as the number of users increases due to the Birthday Problem, but we're still in the 2.9 x 10^39 range with a billion addresses).

http://www.hawaii.edu/suremath/jsand.html

6. Double Spending in Bitcoin

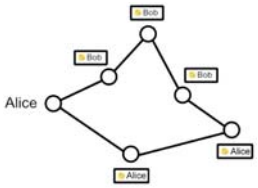
Let's recap Bitcoin security so far. By verifying the Digital Signature, we know that only the true owner could have created the transaction message. And to make sure the sender actually has money to spend, we also check each referenced input, making sure it is unspent. But there is still one large security hole in the system that can make this "unspent check" unreliable, and this has to do with the **order** of transactions.



Considering that transactions are passed node-by-node through the

network, there's no guarantee that the order in which you receive them represents the order in which they were created. And you shouldn't trust a timestamp because one could easily lie about the time a transaction was created. (Contrast this with a centralized system like paypal, where it's easy for a central computer to keep track of the order of transactions.)

Therefore, you have no way to tell whether one transaction came before another, and this opens up the potential for fraud. A malicious user, Alice, could send a transaction giving money to Bob, wait for Bob to ship a product, and then send another transaction referencing the same "input" back to herself.

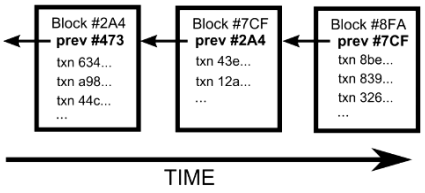


Because of differences in propagation times, some nodes on the network would receive the 2nd "double-spending" transaction before the one to Bob. And when Bob's transaction arrived, they would consider it invalid because it's trying to re-use an input. So Bob would be out both his shipped product and his money. Overall, there would be disagreement across the network about whether Bob or Alice had the money, because there's no way to prove which transaction came first.

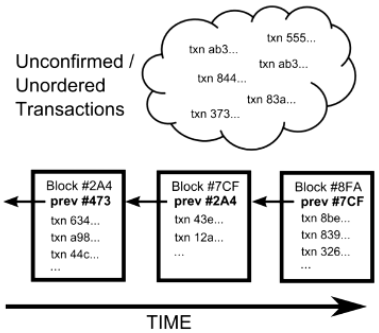
In light of this, there needs to be a way for the entire network to agree about the order of transactions, which is very much a daunting challenge in a decentralized system. Bitcoin's solution is a clever way to both determine and safeguard the ordering through a kind of mathematical race.

7. The Block Chain: an Ordering of Transactions

The **Bitcoin** system orders transactions by placing them in groups called blocks, and linking those blocks together in something called the **block chain**. Note that this is different from the transaction chain we discussed earlier. The block chain is used to order transaction, whereas the transaction chain keeps track of how ownership changes.



Each block has a reference to the previous block, and this is what places one block after another in time. You can traverses the references backwards all the way to the very first group of transactions ever made. Transactions in the same block are considered to have happened at the same time, and transactions not yet in a block are called "unconfirmed," or unordered.



Any node can collect a set of unconfirmed transactions into a block, and broadcast it to the rest of the network as a suggestion for what the next block in the chain should be. Because multiple people could create blocks at the same time, there could be several options to choose from, so how does the network decide which should be next? We can't rely on the order that blocks arrive, because, as explained with transactions above, they may arrive in different orders at different points in the network.

Part of Bitcoin's solution is that each valid block must contain the answer to a very special mathematical problem. Computers run the entire text of a block plus an additional random guess through something called a **cryptographic hash** until the output is below a certain threshold.

A hash function creates a short digest from any arbitrary length of text, in our case, the result is a 32 byte number. Here are some examples of the specific hash function Bitcoin uses, SHA256:

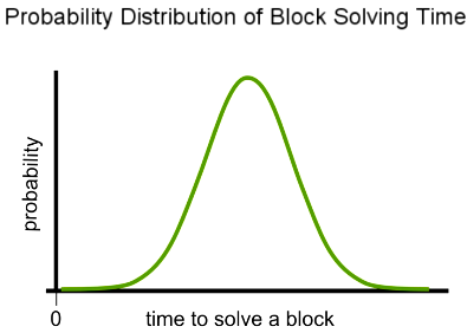
SHA256("short sentence"): 0x 0acdf28f4e8b00b399d89ca51f07fef34708e729ae15e85429c5b0f403295cc9

SHA256("The quick brown fox jumps over the lazy dog"): 0x d7a8fbb307d7809469ca9abcb0082e4f8d5651e46d3cdb762d02d0bf37c9e592
SHA256("The quick brown fox jumps over the lazy dog.") (extra period added): 0x ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c

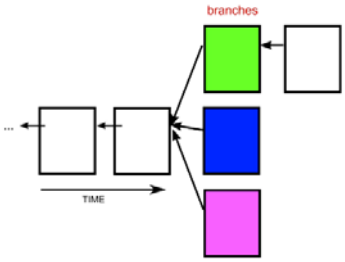
Note how much the output changes in result of a single extra period at the end of the third example. The output is completely unpredictable, so the only way to find a particular output value is to make random guesses. It's very much like guessing the combination to a lock. You might get lucky on your first guess, but on average, it takes many guesses. In fact, it would take a typical computer several years of guessing to solve a block.

With every computer in the entire Bitcoin network all guessing numbers, it takes about 10 minutes on average for someone to find a solution.

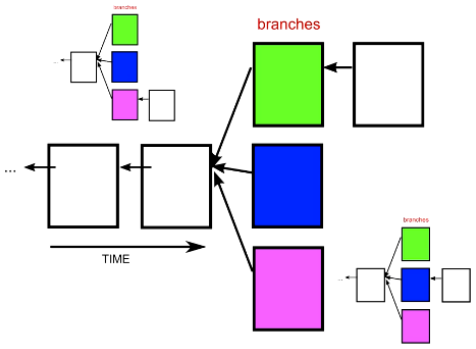
The first person to solve the math problem broadcasts their block, and gets to have their group of transactions accepted as next in the chain. The randomness in the math problem effectively spreads out when people find a solution, making it unlikely that two people will solve it at the same time.



Occasionally, however, more than one block will be solved at the same time, leading to several possible **branches.**



In this case, you simply build on top of the first one you received. Others may have received the blocks in a different order, and will be building on the first block they received.



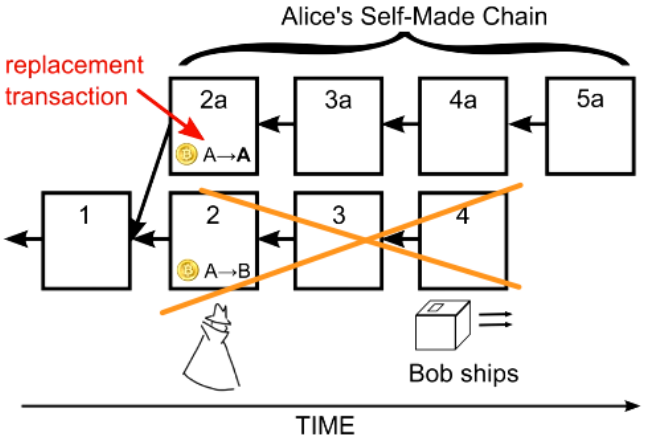
The tie gets broken when someone solves another block. The general rule is that you always immediately switch to the longest branch available. The math makes it rare for blocks to be solved at the same time, and even more rare for this to happen multiple times in a row. The end result is that the block chain quickly stabilizes, meaning that everyone is in agreement about the ordering of blocks a few back from the end of the chain.

8. Double Spending in the Block Chain

The fact that there's some ambiguity in the end of the chain has some important implications for transaction security. For instance, if your transaction finds itself in one of the shorter branches, it will lose its place in line within the block chain. Typically, this means it will just go back into the pool of unconfirmed transactions, and be included in a later block. Unfortunately, this potential for transactions to lose their place opens the door to the very double-spend attack that was our original motivation for an ordering system.

Let's look at how a double spend attack would work in the system described so far. A fraudster, Alice, sends money to Bob. Bob then waits for the transaction to get "confirmed" into the block chain, and then ships a product.

Now, because nodes always switch to a longer branch, if Alice can generate a longer branch that replaces the transaction to Bob with one to someone else, his money will effectively get erased.



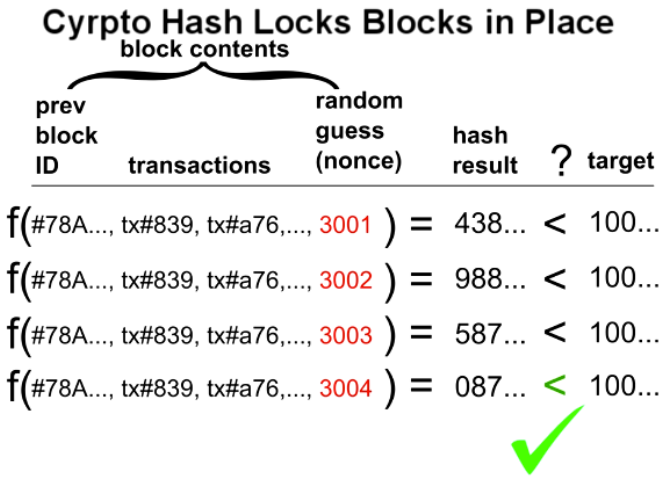
Bob's transaction will initially get tossed back into the unconfirmed pool. But since Alice has replaced it with another transaction that uses its same input, nodes will now consider Bob's transaction invalid, because it's referencing an already spent input.

9. Double Spend Prevention

So how does the ordering system prevent Alice from defrauding Bob? You might think that Alice could pre-compute a chain of blocks to spring on

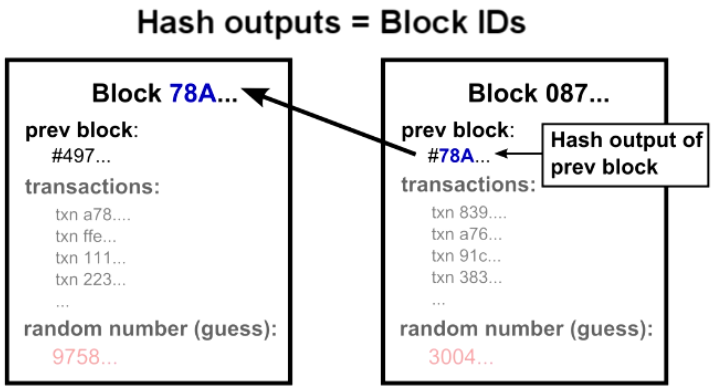
the network at just the right time, but the math puzzles in each block actually prevent this. We need to look a little deeper into the cryptographic hash explained earlier to fully understand why.

As mentioned previously, solving a block involves trying to get the cryptographic hash of the block to be below a certain value, and you do that by trying different random numbers at the end of the block. Once solved, the hash output is like a fingerprint that uniquely identifies that block. If even a single character in the block is changed, the block's hash would be completely different, just like we saw before when an additional period was added.

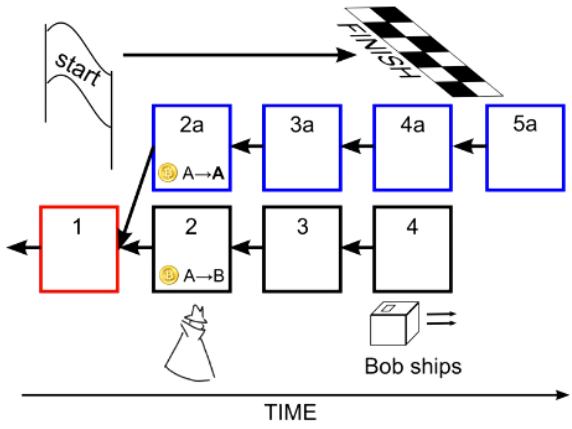


The hash output, or fingerprint, is actually what's used as the "previous block" reference. One result of this is that there's no way to switch out a block in the middle of the chain, because the hash value for the new block would be different, and the next's block reference would no longer point to it. And subtly, but even more importantly, a block cannot be solved before the previous block is solved.

The previous block reference is part of the text that goes through the hash function, so any changes to it would require resolving.



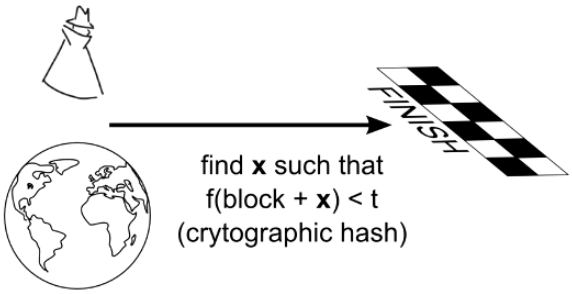
Getting back to Alice, this is why she can't precompute a branch. She can only start solving blocks once the block she wants to build on is solved, and its hash value is known. She is therefore in a race with the rest of the network until Bob ships a product, which is when she wants to present a longer branch. She must work in private, because if Bob heard about her double spend block, he would obviously not ship the product.



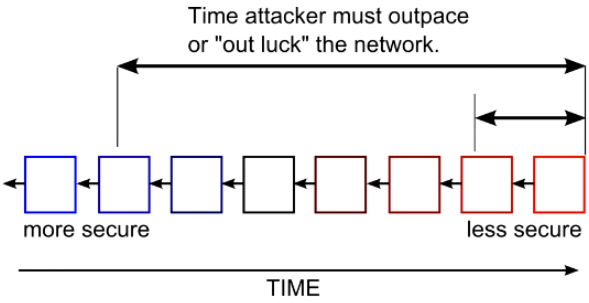
One last question is whether Alice might be able to outpace everyone if she had an extremely fast computer, or perhaps a room full of computers. But even with thousands of computers, she would be unlikely to win the race

to solve a block, because she isn't racing any one computer, but rather the entire network. You can think of it like a lottery. She can operate thousands of computers, or equivalently, buy thousands of lottery tickets, but even then, it's much more likely that someone else will win. She would need to control half of the total computing power in the entire network to have a 50% chance of solving a block before someone else. And much more to have a high probability of winning several blocks in a row faster.

Transaction Order protected by Race



So transactions in the block chain are protected by a mathematical race--one that pits an attacker against the entire rest of the network. A consequence of blocks building on top of each other is that transactions further back in the chain are more secure. An attacker would have to outpace the network for a longer amount of time to carry out a double spend attack, and replace a block. So the system is only vulnerable to a double spend attack near the end of the chain, which is why it's recommended to wait for several blocks before considering received money final.



One last comment on the block chain before explaining the final pieces of the Bitcoin system. Amazingly, nothing described so far requires any trust. When you receive information from strangers in the Bitcoin network, you can check for yourself that the block solutions are correct. And because the math problems are so hard, you know that there's no way any attacker could have generated them on their own. The solutions are proof that the computing power of the entire network was brought to bear.

10. Mining and Pools

Now that we've discussed how money is transferred through Digital Signatures and transaction chains, and how the order of those transactions is protected in the Block Chain, let's go over the final piece: where Bitcoins come from. To send money, you must reference a previous transaction where you were the recipient, but how do coins get into this ownership chain in the first place?

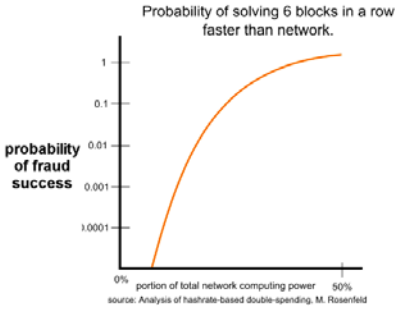
As a way to slowly and randomly generate and distribute coins, a "reward" is given to whoever solves a block. This is why solving blocks is called mining, although its real purpose is to verify transactions, and

safeguard the block chain. Every 4 years, the block reward is cut in half, so eventually no more coins will be released--about 21 million in total will be created. Bear in mind that you can send down to 1 / 100 millionth of a Bitcoin (.00000001), so the total number available will likely not limit the currency's usability.

Once the block rewards cease, what incentive will miners have to process transactions? In addition to the block reward, miners also get any transaction fees that can optionally be included with transactions. Right now, miners will include transactions with no fees into blocks because their main incentive is the block reward, but in the future, transactions will likely be processed in order of the fees attached, and ones without fees will likely be ignored. So sending money in Bitcoin will probably not be free, but will hopefully still be cheaper than current credit card fees.

11. Mining Pools

As mentioned before, on average, it would take several years for a typical computer to solve a block, so an individual's chance of ever solving one before the rest of the network, which typically takes 10 minutes, is very low. To receive a steadier stream of income, many people join groups called mining pools that collectively work to solve blocks, and distribute rewards based on work contributed. These act somewhat like lottery pools among co-workers, except that some of these pools are quite large, and comprise more than 20% of all the computers in the network.



The fact that some of these pools are so large has some important implications about security. As mentioned before, it's very unlikely for an attacker to solve several blocks in a row faster than the rest of the network, but it is possible, and the probability increases as the attacker's processing power gains in proportion to the rest of the network. In fact, one of the mining pools, BTC Guild, has solved 6 blocks in a row by itself, and has voluntarily limited its members to ward off distrust of the entire bitcoin network.

Even with substantial computing power, the farther back in the block chain a transaction gets, the harder it would be for an attacker to change it, as they must outpace the rest of the network for the time between when a transaction is sent, and when a product is shipped.

The current recommendation is to wait for a transaction to make it into at least one block, or get one confirmation, before considering it final. And for larger transactions, wait for at least 6 blocks. In light of BTC Guild's ability to solve 6 blocks in a row, you might want to wait even longer.

12. Confirmation Time

By design, each block takes about 10 minutes to solve, so waiting for 6 blocks would take about an hour. Compared to the several seconds a credit card transaction takes, waiting this long for a confirmation may seem burdensome, but keep in mind that credit card customers can claim a stolen card months later to have charges reversed from merchants (called charge backs), so Bitcoin is actually much faster from a merchant's perspective.

Why 10 minutes per block? The particular choice of 10 minutes was somewhat arbitrary, but extremely short times could lead to instability, and longer ones would delay confirmations. As more computers join the network, and specialized hardware is designed specifically for mining, the block solution time would get very small. To compensate, every two weeks, all the Bitcoin software recalibrates the difficulty of the math problem to target 10 minutes. For comparison, a similar digital currency called Litecoin has been able to operate with a 2.5 minute block time.* *a paper by M. Rosenfeld, Analysis of hashrate-based double-spending, concludes that security is a function of the number of blocks, and not the time used to solve each block, but this assumes an attacker's computing power is not dependent on time, ie, he could overpower the network for days just as easily as a few minutes.

*also see comments by Satoshi in this forum post regarding block time and system efficiency: https://bitcointalk.org/index.php?topic=130222.60

13. Conclusion and Summary of How Bitcoin Works

In summary, Bitcoin is a mathematically protected digital currency that is maintained by a network of peers. Digital Signatures authorize individual transactions, ownership is passed via transaction chains, and the ordering of those transactions protected in the Block Chain. By requiring difficult math problems to be solved with each block, would-be attackers are pitted against the entire rest of network in a computational race they are unlikely to win.

Bitcoin promises many interesting ideas, such as insulation from government meddling, anonymity, and potentially lower transaction fees. It also has many challenges, as it is currently very difficult to exchange Bitcoins for other currencies, and it has been cited as a haven for illegal activity and tax evasion, so governments may try to ban it. Also, the mathematical race that protects the Block Chain uses a substantial amount of electricity.

14. Bonus: Digital Signature Math

As explained above, a Digital Signature allows me to prove I have a private key, and that I used that private key to sign a particular message, without ever revealing my private key. Here is a simplified method, that, although insecure, shares some of the structure

of a real Digital Signature Algorithm: RSA.

Consider the following variables:

p = public key

q = private key

p*q = N also publicly shared

m = message

if I make a signature from the private key and message, say, s = q*m, then a third party can verify the signature is mine by checking s*p = m*N. This is because

m*q*p = m*N, and N = q*p.

In this way, the third party only needed to know the publicly available public_key, N, message and signature. The only downside of this technique is that someone who knows how to divide would be able to calculate my private key by q = N / p. The real math falls under the umbrella of abstract algebra, and involves modular arithmetic, and our inability to factor extremely large numbers. Consider N = q*p, where q and p are large primes (300+ digits). Given only N, finding q and p would require vast amount of guessing and checking, but knowing q beforehand allows you to find p with simple division. This factorization trap door is what enables the RSA algorithm. The public key / private key algorithm used in Bitcoin is called Elliptic Curve DSA, and is based on the difficulty of finding a discrete logarithm. ECDSA's advantage over RSA is that the key size required for a given amount of security is much smaller.

Contents