

Note: I in no way own any of the knowledge I have written in this document. This is just my one stop reference when I need it. I have made this as a part of learning blockchain technology. I am following the blockchain at Berkeley's course as a base along with many other references wherever I get stuck. Thanks to all the people in the references who've helped me learn blockchain.

What is Bitcoin?

Cryptocurrency:

Completely digital, decentralised currency built on principles of computer science, cryptography and economics

Bitcoin:

Blockchain is the underlying data structure of bitcoin.

The bitcoin blockchain stores the permanent history of all transactions that ever occur in the history of bitcoin. Append-only ledger. It's a shift towards privacy and decentralization. Bitcoin was inspired by the Cypherpunk movement. No real identities of users can be revealed. Users are just addresses, random strings of letters and numbers. There is no third party dependency. Users have the freedom to transact by themselves thus protecting privacy. Although there are no central parties to detect and remove malicious users.

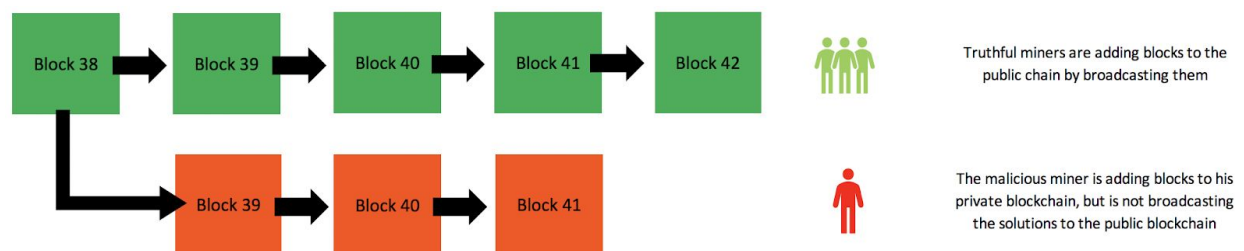
Double Spending attack:

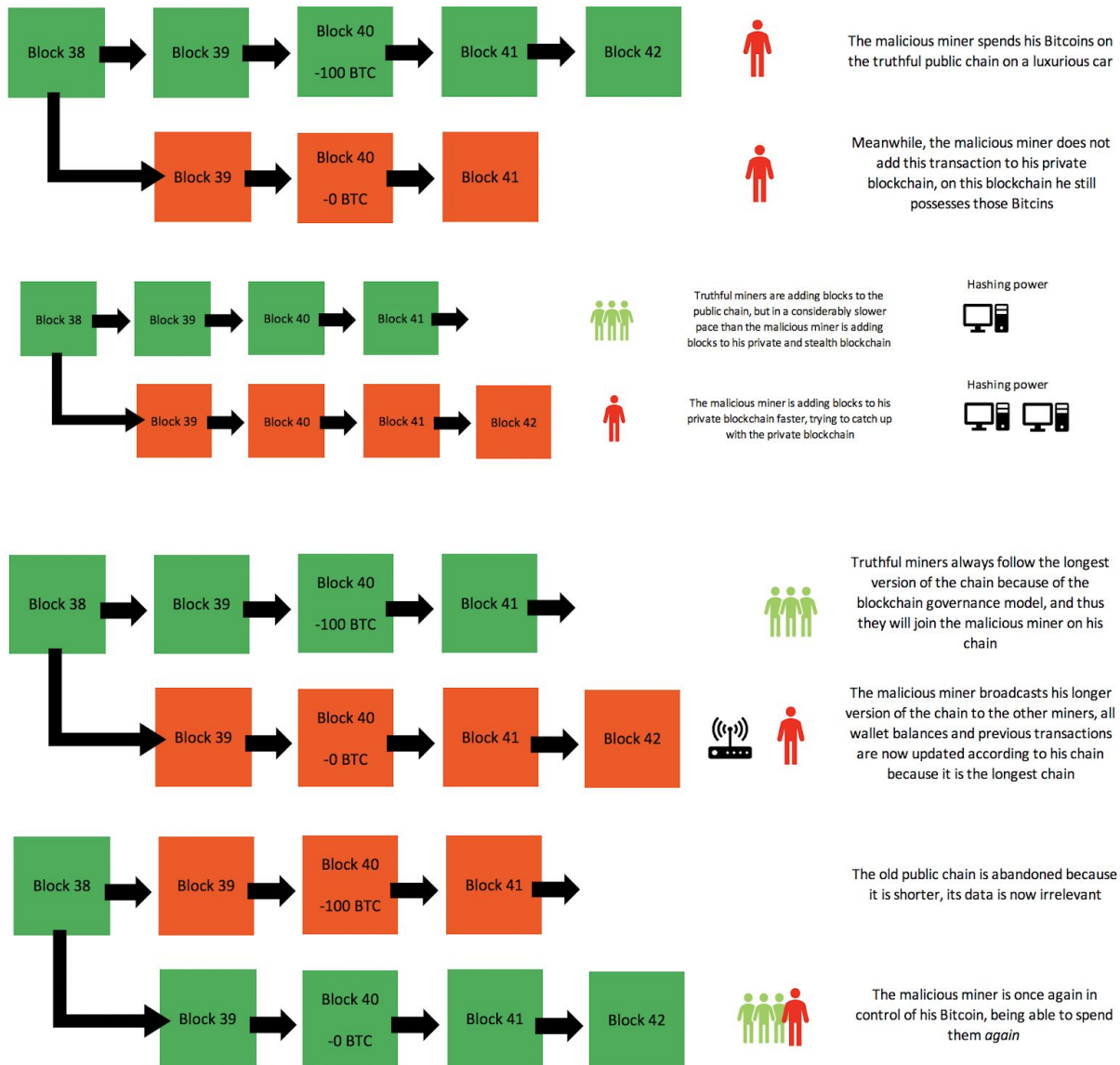
What's that?

Double-spending problem is the successful use of the same funds twice.

let's say I spend 10 Bitcoin on a luxurious car. The car gets delivered a few days later, and my Bitcoins are transferred from me to the car company. By performing a 51% attack on the Bitcoin blockchain, I can now try to reverse this Bitcoin transfer. If I succeed, I will possess both the luxurious car and my Bitcoins, allowing me to spend those Bitcoins again.

<https://blog.goodaudience.com/what-is-a-51-attack-or-double-spend-attack-aa108db63474>





Cypherpunks:

They advocate for the protection of privacy using cryptography. They don't trust government, etc. These centralization points gather huge amounts of information of users.

Bitcoin Network:

It is a group of users communicating with each other as a part of bitcoin protocol. This n/w validates transactions before storing them in the transaction history.

Satoshi Nakamoto published the bitcoin white paper.

Explaining blockchain to a five year old Alice:

Alice, you have a lot of friends in your society right? Yep, so assume your friend Bob asks you to lend him a few toys which he will return to you in a few days. But since those are your favourite toys,

you ask him to sign on a piece of paper. He does. After few days, when you ask him to return, he denies. He says he doesn't know about the sign or toys. What do you do?

Now assume this happens among all your friends. Seeing this, jenny thinks to maintain a notebook, where every transaction that will take place will take place in the club house and will be entered in the notebook. Looks good. Everybody gives or takes toys like that. But one day, the notebook was destroyed because someone spilled milk on it by mistake. Every transaction is lost! Problem that databases face. A single point of failure.

Jenny comes up with a better idea. A distributed database. She chooses some trustworthy people and distributes the notebooks amongst them. After every transaction, jenny informs them to write in their notebooks as well. One day, raj, jenny's best friend had borrowed a lot of chocolates and toys and he broke most of them. He asks jenny a favour. To remove all his transactions and so she did. This is the problem with distributed database, they are centralized. Now when everybody got to know about this:



Every child decides to keep a notebook. Every time a transaction is made all the n people write it in their books, and no person controls the overall representation of the transactions. This is decentralization.

They never remove any previous transaction. This is immutability. Suppose some mischievous children try to cheat and say wrong transaction to everybody, but since now everybody has a note, they claim this a fraudulent transaction and banish c from further adding any data. They don't add what c said on the consensus protocol, where voting is done on the basis of validity of transaction in the blockchain.

Since these transactions form a chain, this fully decentralized, immutable notebooks are called block chain.

Bitcoin vs. Banks

Instead of trusting people in suits, we should trust the math and logic.

Account and Identity Management	Service	Record Management	Trust
 Links personal information to bank account and verifies ownership	Transfers money and redeems money	Updates and tracks account balance	Provides services by professionals under regulations of government
Account and Identity Management	Service	Record Management	Trust
 Gives users autonomously created and managed identities	Sends funds between peers directly (P2P)	Updates every node, which keeps its own ledger (blockchain)	Provides trusted protocol which incentivizes actors to behave honestly

Stage-1 - Identity:

An identity helps in authentication process. Your money should solely be yours and the decisions of spending, receiving money should be known to you and by your permission. If someone tries to cheat and take away the money, you should know who they are in order to blacklist them. There should also be integrity. If there's supposed to be a transaction of 10 bitcoins, no one should be able to tamper it to 100, etc.

emails have aliases and **passwords**, bitcoins have public keys and **private keys**. You would give your email/public keys to others so that they know it's you. And you could access your accounts through the private keys/passwords to know that you have the ownership of that public key. Other people send something to the address. Address is derived from the public keys. Users have to generate their own identity, there's no central authority to do that. Private keys are chosen at random. And public keys are generated from private keys using a mathematical function.

What if two people get the same private key?

Let's consider no. of possible address bitcoin has to the grains of sand on earth. The likelihood of two people picking the same grain of sand is less than 0.0001%. No. of bitcoin addresses is many more in magnitude, making the probability smaller. Now, For every grain, consider there is another earth again with that many grains of sand. Refer the following:

“What if someone guesses my private key?!”

- Bitcoin is hidden in the large amount of public keys
 - 2^{160}
(1,461,501,637,330,902,918,203,684,832,716,283,019,655,932,542,97) possible addresses
- Practically impossible for anyone to overlap using random generation of public key
 - For reference:
 - Grains of sand on earth: 2^{63}
 - With 2^{63} earths, each with 2^{63} grains of sand: 2^{126} total grains of sand
 - 2^{126} is only **0.0000000058%** of 2^{160}
 - Population of world: 7.5 billion in April 2017
 - Every person could have about 2^{127} addresses *all to themselves*

Stage 2 - Transactions

What makes a transaction valid?

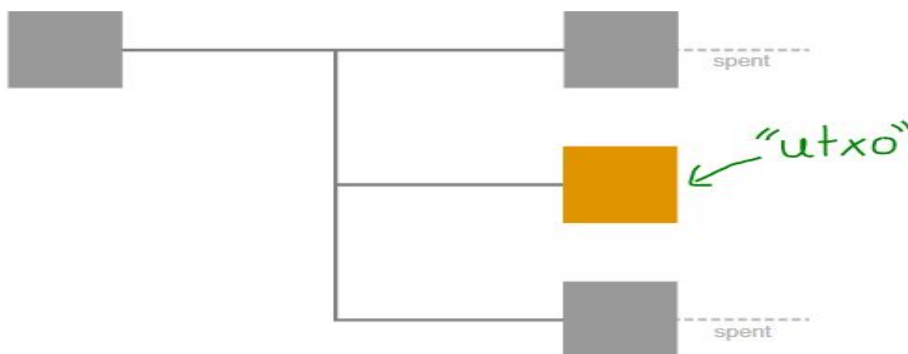
- Proof of ownership (signature)
- Available funds
- No other transaction using the same available funds

Two types of **record-keeping models** are popular in today's blockchain networks. The first method is called the **UTXO (Unspent Transaction Output) Model** and the second one is the **Account/Balance Model**. *The UTXO model is employed by Bitcoin, and Ethereum uses the Account/Balance Model*

UTXO :

After an output has been “used up” in a transaction, it cannot be used again. So a spent output is like a corpse.

Unspent outputs, however, are alive. They are available to be used in new transactions (as long as you can unlock them), which makes them useful. That's why there is a distinction between spent outputs and unspent outputs (UTXOs). Users do not spend from an account, they spend directly from transactions made to them.



When you look at your Bitcoin wallet, you see a balance. For this example, let's set that at 100 bitcoin. Although you observe just one balance, your funds are actually comprised of several UTXOs. You may have four UTXOs worth 25 bitcoin each, two UTXOs worth 50, or a set of UTXOs valuing 37, 18, 40, and 5 bitcoin. The specific amounts don't matter, but they must add up to your total balance, in this case, 100.

Furthering the example, let's assume that you're shopping around for a new car. You work hard; you deserve it. Bucking the Lambo stereotype, you decide on a Porche that costs 35 bitcoin. Well, your wallet only contains UTXOs equaling 15, 17, 28, and 40 bitcoin each. You don't have one valued at precisely 35 bitcoin.

It's impossible to split UTXOs, so there's no way to pay the exact 35 bitcoin that you owe.

Instead, you spend the 40 bitcoin UTXO. In its place, the network mints two new UTXOs: one valued at 35 bitcoin, one worth 5 bitcoin. The car dealership receives the 35 bitcoin UTXO while you receive the 5 bitcoin UTXO as change.

You may also spend the 17 and 28 bitcoin UTXOs and receive 10 bitcoin as your change. A transaction may use any combination of UTXOs; however, you don't have control over which ones.

Transaction fees are also included in transactions and subtracted from the UTXO that you receive as change. The equation looks something like this:

New UTXO = (Sum of UTXOs in the transaction) – (Transaction amount) – (Transaction fee)

Continuing our most recent example with a one bitcoin transaction fee:

New UTXO = (17+18) – (35) – (1) = 9 bitcoin

Let's take a look at a simplified example of how the UTXO model works in Bitcoin transactions:

1. Alice gains 12.5 bitcoins through mining. Alice's wallet is associated with one UTXO record of 12.5 bitcoins.
2. Alice wants to give Bob 1 bitcoin. Alice's wallet first unlocks her UTXO of 12.5 bitcoins and uses this whole 12.5 bitcoins as input to the transaction. This transaction sends 1 bitcoin to Bob's address and the remainder of 11.5 bitcoins is sent back to Alice in the form of a new UTXO to a newly-created address (owned by Alice). This is known as the change UTXO.

3. Say there was another UTXO of 2 bitcoins associated with Bob prior to step 2, Bob's wallet now shows that his balance is 3 bitcoins. Bob's wallet now keeps track of two UTXOs: one from before and another from the transaction in step 2. Each UTXO needs to be unlocked if Bob wishes to spend them.

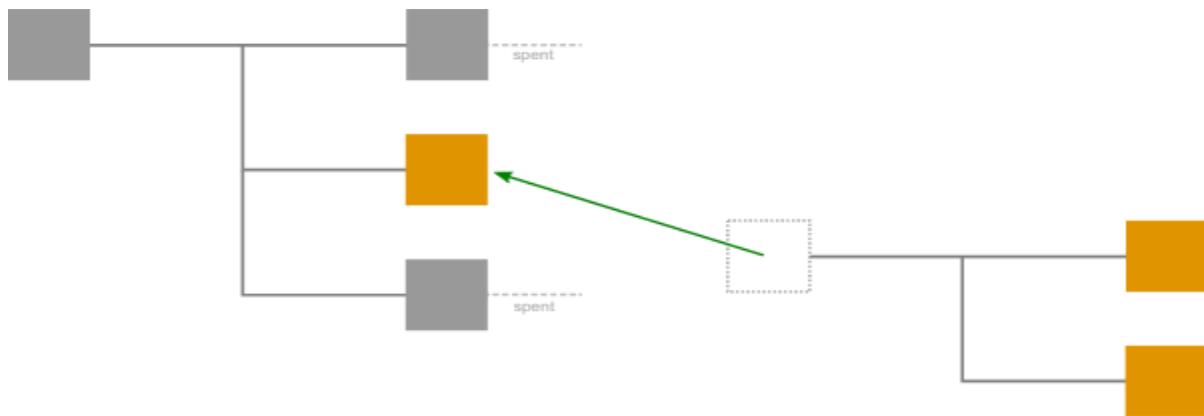
Advantages of UTXO :

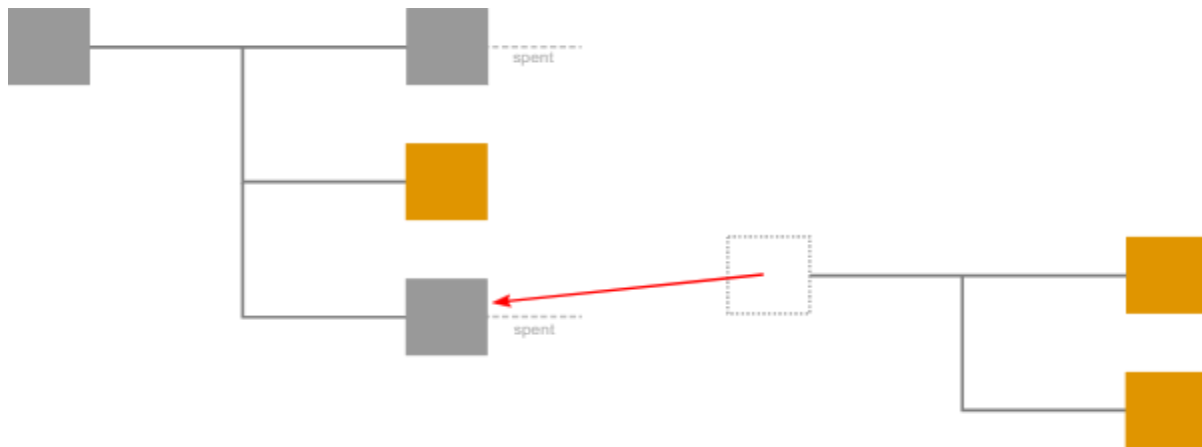
- Simplicity. Spending a UTXO is all or nothing. Since UTXOs are uniquely referenced and completely consumed upon spending, there is no chance for a transaction to be replayed.
- Privacy preserving behavior is encouraged in the UTXO model. Users are encouraged to generate a new address for every incoming transaction including change addresses. By using a new address each time, it is difficult to definitively link different coins to a single owner.

UTXOs are used in verifying transactions:

A node will verify the transactions it receives by checking that its inputs have not already been spent. So if you want to create your own bitcoin transaction, you must use UTXOs in your inputs:

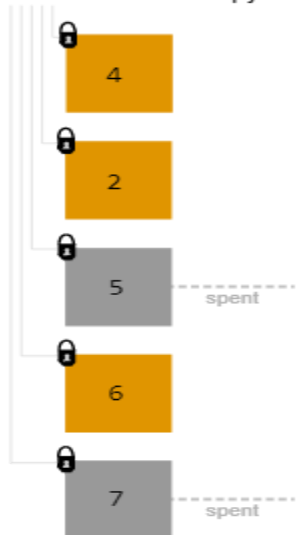
However, if you try and use an output that has already been used in another transaction, your transaction will be rejected by nodes.





They can also be used in knowing address balances. If you want to work out the balance of an address, add up all of the unspent outputs that are locked to that address.

1addressLv4r8xpj4cTvnLMAB4FoorFAQx



balance = 12 BTC

Because UTXOs are needed to verify every transaction your node receives, the UTXOs are stored in their own database.

~/.bitcoin/chainstate/

This database allows your node to quickly verify transactions, by checking to see if the transaction's inputs are available in the UTXO database. The UTXO database is also loaded into RAM when you run bitcoind, which further helps to speed up verification.

The Account/Balance Model, on the other hand, keeps track of the balance of each account as a global state. The balance of the account is checked to make sure it is larger than or equal to the spending transaction amount.

Here is a simplified example of how this model works in Ethereum:

1. Alice gains 5 ethers through mining. It is recorded in the system that Alice has 5 ethers.
2. Alice wants to give Bob 1 ether, so the system will first deduct 1 ether from Alice's account, so Alice now has 4 ethers.
3. The system then increases Bob's account by 1 ether. The system knows that Bob has 2 ethers to begin with, therefore Bob's balance is increased to 3 ethers.

Advantages of Transaction models :

- Large space savings: for example, if an account has 5 UTXO, then switching from a UTXO model to an account model would reduce the space requirements from $(20 + 32 + 8) * 5 = 300$ bytes (20 for the address, 32 for the txid and 8 for the value) to $20 + 8 + 2 = 30$ bytes (20 for the address, 8 for the value, 2 for a nonce).

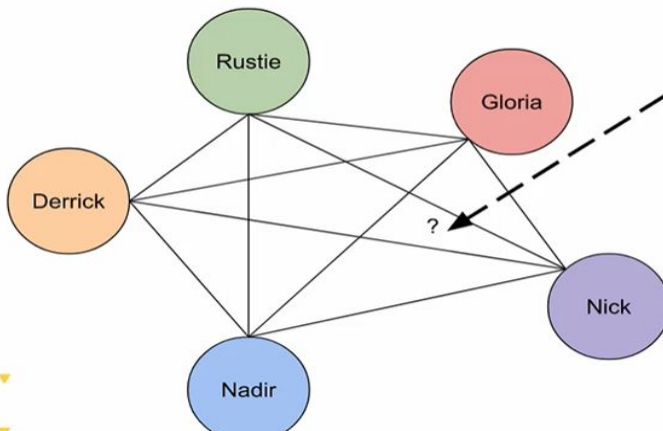
In reality savings are not nearly this massive because accounts need to be stored in a Patricia tree but they are nevertheless large. Additionally, transactions can be smaller (eg. 100 bytes in Ethereum vs. 200-250 bytes in Bitcoin) because every transaction need only make one reference and one signature and produces one output.

References: <https://coincentral.com/utxo-beginners-explainer/>
<https://learnmeabitcoin.com/guide/utxo>
<https://medium.com/@sunflora98/utxo-vs-account-balance-model-5e6470f4e0cf>
<https://medium.com/@jcliff/intro-to-blockchain-utxo-vs-account-based-89b9a01cd4f5>
<https://github.com/ethereum/wiki/wiki/Design-Rationale#accounts-and-not-utxos>

Stage 3 - Record Keeping:



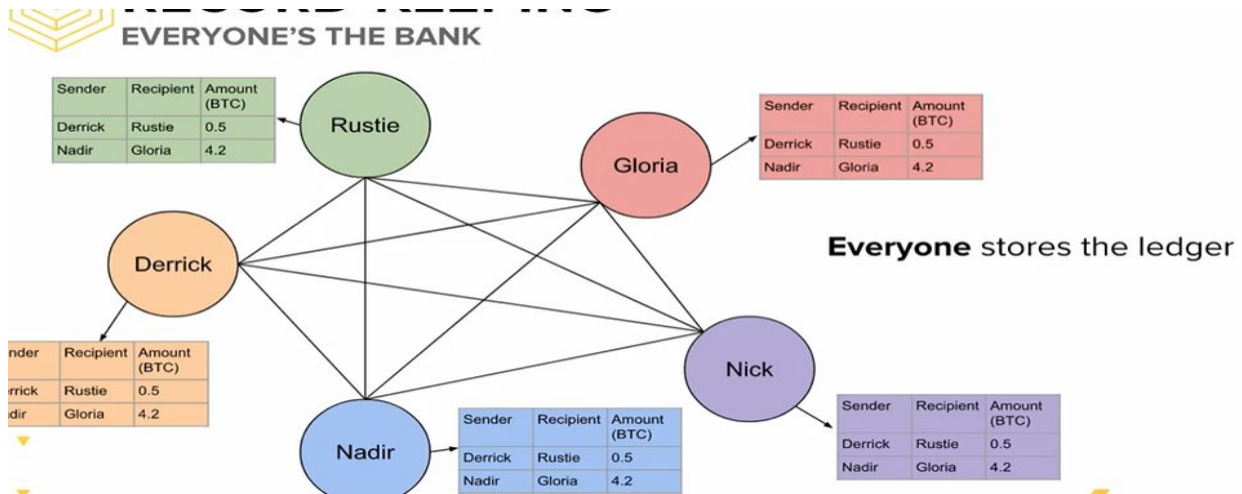
RECORD-KEEPING DISTRIBUTED DATABASES



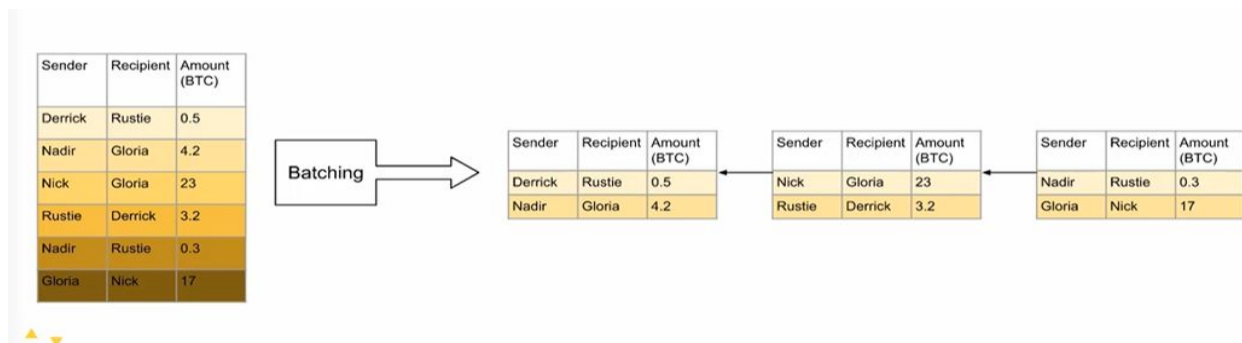
Sender	Recipient	Amount (BTC)
Derrick	Nadir	0.5
Rustie	Gloria	4.2

We know how to represent identities and transactions---how do we store all that information? How do we keep track of this ledger of transactions?

⇒ With a **distributed database**



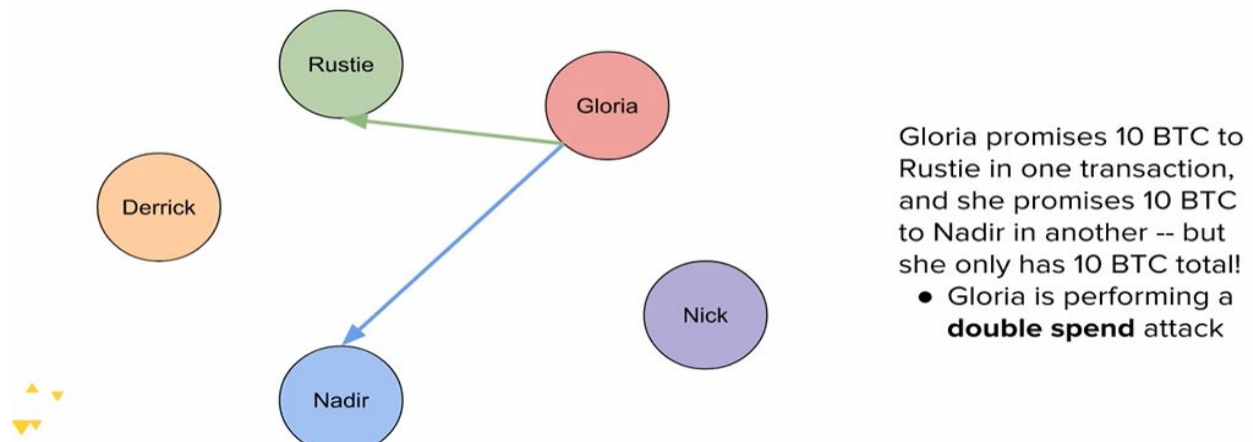
We might naively decide to store every transaction individually, but for a network that may be dealing with many transactions per second, updating the database for every received transaction will be costly, especially since this update would have to be delivered to everyone in the network. Everyone maintains their own ledger after all, and once a change is made for one entity, it must propagate throughout the entire network. Thus we use a data structure called blockchain. Here whenever an update is done to the transaction history, a new block is made and appended to the list where all the previous blocks represent history of transaction. Each block has reference of the previous block. Thus mutating anything in the past will affect all the blocks ahead of that tampered block making blockchain tamper evident.



Stage 4 - Consensus:

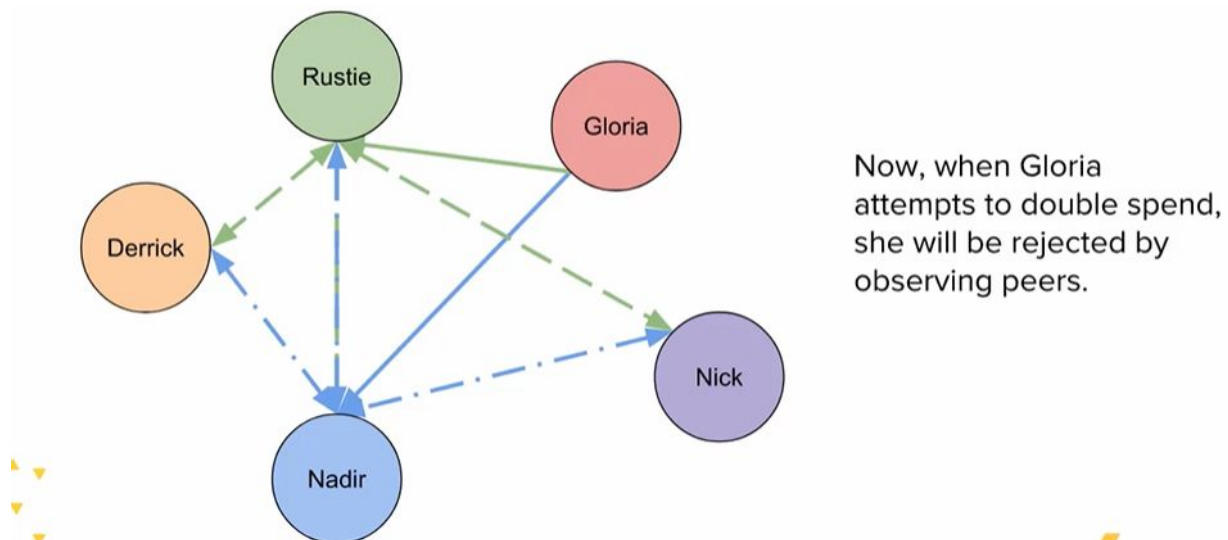
How do we make updates to the blockchain. How do we decide which transactions are valid? In bitcoin, users on the network must come to consensus or agreement on the next valid update.

Let's first say any transaction taking place is known with certainty to the sender and the receiver only and all others are just informed about the transaction. However this may lead to the double spend attack problem.



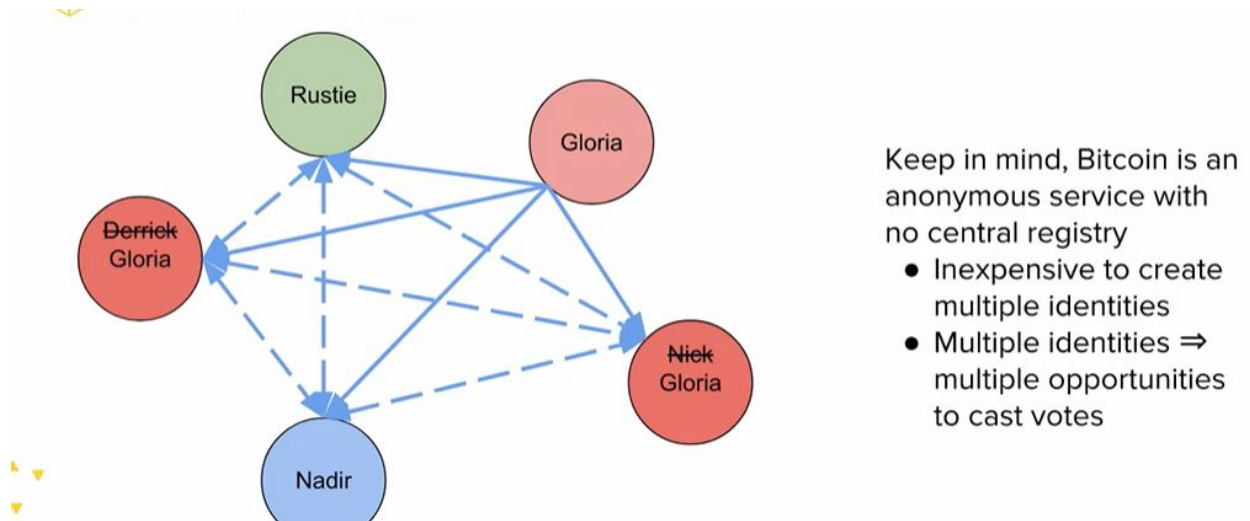
Derrick and Nick know nothing about either transaction. Both Nadir and Rustie feel the transaction is valid and give their laptops to gloria. However when they both try to redeem the bitcoins, the issue becomes transparent. Derrick and Nick have no idea of the transaction, thus they won't help. In this scheme where entities only see the transactions that directly involve them as sender or recipient, it is impossible to come to consensus on a history of transactions because of dishonest actors.

We can set up a voting system instead. One person at a time makes a proposal about an update, and everyone else votes on whether or not to accept the proposal. The person who wants to make a transaction sends the transaction to everyone in the entire network, not just the recipient of bitcoins. Everyone on the network then casts votes based on whether the transaction they saw was valid or not. Only after receiving a certain number of votes, say a majority, does the transaction get saved.



Communication helped here.

However, gloria can make her identities as many as she wants due to the anonymous nature of bitcoin and get the majority of upvotes.



This is called a **sybil attack**, where user creates multiple identities for some malicious purpose. With this votes become meaningless.

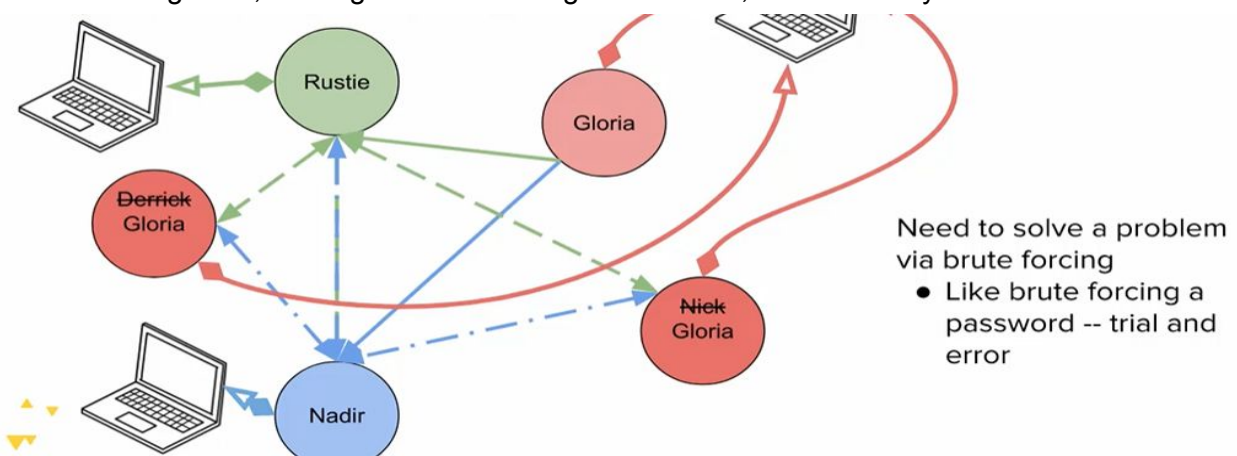
To ensure that every real person has one vote, we make voting expensive. Pay to Play!

Instead of casting votes with identities, we cast votes with resources. In the case of bitcoin, it's computing power.

In satoshi's whitepaper, he envisioned a **1-cpu-1-vote** network instead of 1-identity-1-vote. Proof of work consensus algorithm. By which users provide evidence of spending resources.

Whenever someone wants to make a proposal to the rest of the network, they need to solve a computationally difficult problem by doing a significant amount of work. This problem is uniquely generated based on the information within the proposed block and cannot be predicted beforehand. Trial and error (once we have done before).

In the following case, although there are 5 digital identities, there are only 3 real world identities.



Because Nadir and Rustie recognize that there are two transactions trying to spend the same funds, they can vote against both transactions being included in the blockchain. Gloria with her single computer is incapable of outvoting them, which is precisely what is Needed! By tying our voting power to our real world identities through the scarce resource of computing power, we have successfully eliminated Sybil attacks from Bitcoin and solved the double spending problem.

51% Attack:

[Do you REALLY understand Bitcoin 51% Attack? Programmer explains.](#)

What I understood:

- Out of more than one conflicting blockchains, the longest one with the most difficulty is chosen by the network as the trusted one.
- What does it mean if I have the 51% of the hashing power? I am able to mine faster than rest of the miners. If there is a goal to mine as many blocks as possible, I will win as I can calculate proof of work faster as I have most hashing power.
- Can lead to double spending as well - I mine my own blocks privately using my hashing power without announcing it on the network publicly. I also am doing transactions in the public network, buy a house, super computer, etc. Since I am mining faster and in the version of the blockchain that I have created privately I do not add any of the transactions I did in the public network. After a couple of blocks, I announce my version of blockchain in the public, and the network will accept it, because it will be the longest one. Also, I bought the assets but since these transactions are not on the network, I get to spend those bitcoins in my blockchain that is recently accepted.
- Other miners would drop as they go out of business. I would become the monopoly as I am getting all the block reward. No one then would be a part of this monopoly blockchain.

Forking:

[\(How bitcoin/ethereum gets updated?\)](#)

Sometimes, different miners may create different blocks, either intentionally (e.g. double spending) or unintentionally, to add at the same point on the blockchain. This creates multiple chains: multiple different versions of the transaction history. We say that the blocks are competing at the same block height, and that there has been a fork. Following protocol, miners eventually resolve the fork and agree upon one of the chains to be the valid blockchain, and continue to build blocks upon it.

While some forks occur naturally, and some are the result of double spending attempts, there also exist purposeful cases of forking, used to make changes to the Bitcoin protocol.

[How Bitcoin and Ethereum get updated | Programmer explains soft and hard forks](#)

<https://www.binance.vision/blockchain/hard-forks-and-soft-forks>

Soft fork: New version is backward compatible with the old version. That means that non-updated nodes are still able to process transactions and push new blocks to the blockchain, so long as they don't break the new protocol rules. Users who have upgraded can communicate with those who haven't.

Imagine a soft fork which makes a new rule lowering block size from 3MB to 2MB. Older nodes will still be able to process transactions

and push new blocks that are 2MB or less. But if an older node tries to push a block that is greater than 2MB to the network, newer nodes will reject the block because it violates the new rules. That encourages older nodes to update to the new protocol since they aren't as efficient as the updated ones.

Hard fork: Not compatible with the old versions, meaning that nodes that don't update to the new version won't be able to process transactions or push new blocks to the blockchain. Hard forks can be used to change or improve an existing protocol, or even to create a new, independent protocol and blockchain. After the common fork point, they become two completely different blockchains with no communication. Hard fork splits the network, thus everyone is careful during the hard fork. Ethereum hard forked and new version is ethereum and old version is ethereum classic and rolled back the theft of 50 million dollars in the new version.

Imagine a change in a protocol which increases the block size from 2MB to 4MB. If an updated node tries to push a 3MB block to the blockchain, the older, non-updated nodes will not see this block as valid, and they'll reject it.

Depending on the situation, hard forks can either be planned or controversial.

In a planned fork, participants will voluntarily upgrade their software to follow the new rules, leaving the old version behind. The ones who don't update are left mining on the old chain, which very few people will be using.

But if the fork is controversial, meaning that there's a disagreement within the community about the upgrade, the protocol is usually forked into 2 incompatible blockchains - 2 different cryptocurrencies. Both of the blockchains will have their own community, and the developers will progress in the way they believe is the best. Since a fork is based on the original blockchain, all transactions from the original blockchain are also copied into the new fork. For example, if you have 100 coins of a cryptocurrency called Coin A, and a hard fork based on that cryptocurrency creates a new cryptocurrency called Coin B, you'll also get 100 coins of Coin B.

Read these later:

<https://bitcoin.org/en/developer-guide>

<https://www.wired.com/2015/04/silk-road-1/>