

Blockchain

Berkely blockchain course **Lecture 1:Bitcoin protocol and consensus**

<https://www.youtube.com/watch?v=AUFOr9qWnhs&list=PLLkiRvMHnp6OIWkZVa85g2tv7NtlgoAID&index=1>

There are some basic definitions to start with such as:

Cryptocurrency: is a form of currency that is stored completely digitally and isn't issued by a central authority.(it means that cryptocurrency is a kind of money that you can not visibly see and it also is not been produced by a center (like bank))

And as an example of cryptocurrency we can name bitcoin (pay attention that bitcoin is a type of cryptocurrency not the other way around!)

Blockchain: blockchain is a data structure used to represent a cryptocurrency (you can think of it as a kind of complicated database but at core is a place where you store data)blockchains are able to store data in a way that allows multiple parties to come together , look at data and access it reliably without having to trust another.

Now that we have talked about cryptocurrencies we need to know what are the key characteristics of a currency then we can learn more about the new currency we just got familiar with.

- 1.Durability:the currency does not lose value and is not destroyable.(it means that if I have a dollar that dollar does not change its value from one day to another and is sustainable and cannot be destroyed easily)
- 2.portability:The currency is easy to transport from one place to another (I can move it around with me)
- 3.Divisibility:The currency can be easily changed in different denominations.(you can not pay everything with just a 100bills)
- 4.uniformity(fungibility):All units of the currency are identical in value.(no one dollar bill is more worthy than the other)
- 5.limited supply:The supply of the currency can't be arbitrarily inflated.
- 6.Acceptability:The currency must be sufficiently widely accepted.

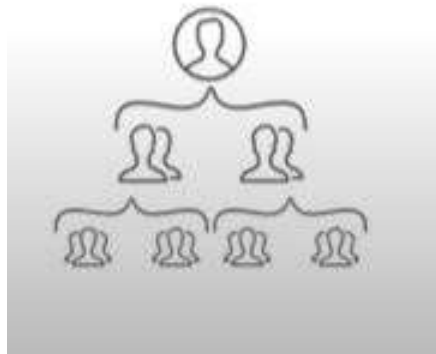
Now we want to talk about the characteristics of a blockchain(just know that you are probably not going to understand them so take it easy) :

- 1.Decentralized control: Communal consensus , rather than one party's decision , dictates who gets to access or update the control of blockchain.
- 2.Tamper-evidence : it's immediately obvious if data stored on the blockchain has been tampered with.(it means that if somebody takes a wrong action every one can see)
- 3.Nakamoto consensus : One has to provably spend resource to proof that you are a real.(The Nakamoto Consensus is a set of rules that verifies the authenticity of a blockchain network, using a combination of the proof-of-work consensus algorithm on a Byzantine Fault Tolerance (BFT) peer-to-peer network.)

So the next definition will be centralization:

It means that the authorization is handled by a **single** party.(data is stored by a single party)

So this is so much like the picture below:



It means that you can use the data but you can not access it .
the owner or the company stores all the data.

Now let's talk about advantages and disadvantages.

Advantages:

1.Efficiency:

Data is stored in one place , programs are executed once.

2.Easy updates:

Updates need one stamp of approval and can be force-pushed to users.(for example if you want to use the facebook you have to do the updates)

Disadvantages:

1.Lack of sovereign:
A central party may choose to use your data arbitrarily.(like facebook sold our data to advertisers and what ever)

2.one point of failure:

Any hack , attack or failure can happen in one place.(all the data is stored in one place)

So the other side of that coin is decentralization.

Decentralization:

Authorization according to an openly-known protocol.

Data is stored by participants.(data is stored by any one who use it)

As we did for centralization we are going to talk about pros and cons for decentralization to.

Pros:

1.sovereignty:

You know exactly how your data will be used.

2.fault tolerance:

The whole network has to get taken down , rather than a single party.

Cons:

1.inefficiency:

Data is duplicated and programs are re-executed across the network.

2.Difficult updates:

Updates must be deliberately adopted by participants in the network.(so instead of facebook just for pushing that update onto all of its users each of its users would have to agree to that update in order for it to be adopted) What is Bitcoin?

It is a cryptocurrency it is the original use of the data structure known as blockchain and it is purely digital and not controlled by any central entity.(blockchain has find a way that if you want to send data from one party to another you do not need the third party)

and it is motivated by cypherpunk movement and that we should fight for our privacy.

As we said earlier about centralization and one party having all the data stored, after the cypherpunk movement people got into thinking that banks are not always reliable and that how can we do the same things that banks do but put it in a decentralized system so that is the first steps of building the blockchain system)

Bitcoin components:

Now that we have talked about the meaning and motivation we are going to talk about bitcoin components and how blockchain works.

- 1.identity: making an account In the system.
- 2.transactions: sending and receiving bitcoin.
- 3.Distributed ledger: recording transaction history.
- 4.trustless consensus: updates to the system and changing to the system.

Identity:

First we need to know why we need an identity:

- 1.for receiving money
- 2.for spending/claiming the money

3.blame: because blockchain is run by a lot of parties if something goes wrong some where or some one makes harm or any thing to the network every individual need to have an ID to know whom to blame.

Identity in bitcoin: Bitcoin has **public keys** (like username in emails) and **private keys** (like passwords in emails).

Public and private keys:

Each identity is represented with a unique public key.a corresponding private key acts as a key to unlock the public key and your money.

Unique private key is generated randomly and public key is derived from private key.(we can always use our private key to connect to our public key but our public key can never be traced back to our private key)

Public key is for receiving, private key is for redeeming.

You should know that for generating a public or private key there is no personal information required.(does that mean bitcoin is anonymous? the answer is no if something were fully anonymous it would mean that there'd be no way to connect me with however I interact with the blockchain network but because we do have this public key this means that it is synonymous and instead there is like a serial code or some way to identify me but it doesn't relate at all to my personal identity)

And also there is no limit about how many accounts you can make. and there is no restrictions on keys that have been taken.

Transactions:

What makes a transaction valid?

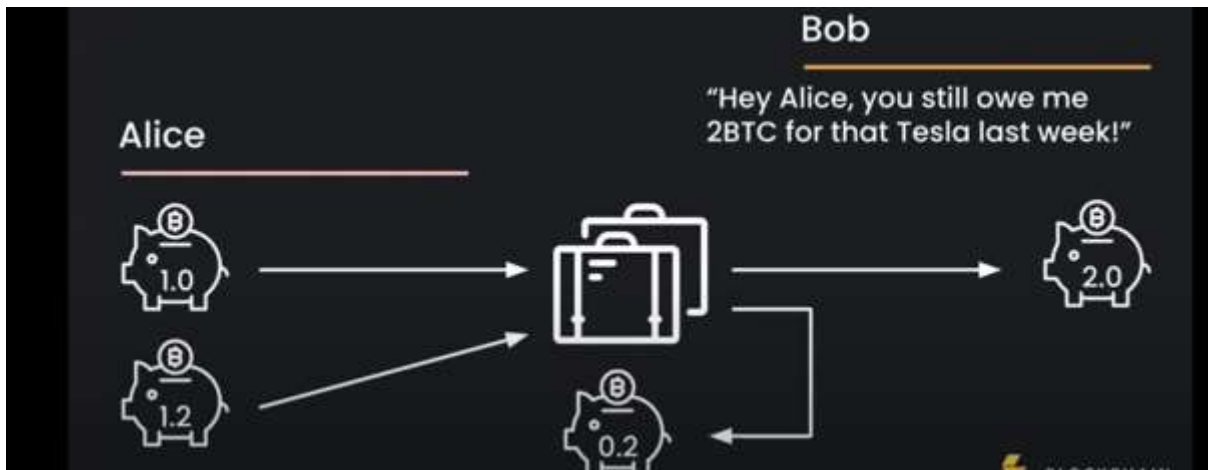
1. proof of ownership: the money you are sending is belong to you.
2. Available funds: you need to prove that you actually have the money.
3. No other transactions using the same funds.

So if we go through the traditional model (what banks do for us in transactions) and build something for our own we can see that in traditional model central manager keeps track of account balances and verifies that transactions are valid it means that they make sure that each account has an available balance and after spending money is subtracted from the total and also after receiving money is added to the total like the picture below:

Alice		Bob	
Balance:	\$100	Balance:	\$250
	-\$10		-\$10

and so in this model it makes sense that you don't need to make sure that those funds aren't being used anywhere else because it's just being subtracted from the balance and once it's subtracted it's gone (automatically goes to bob) but bitcoin is a little different where they actually track each unit of bitcoin whereas here we're looking at balance sheets bitcoin looks at the individual units. so instead of me like sending 50 to someone it would be more like I used these 51 bills and maybe I had those at some point but we need to make sure that those 51 bills weren't used anywhere else when I try to send those to someone and so that becomes a little bit of a problem when you get into bitcoin but when we talk about transactions with bitcoin they use the utxo model utxo stands for *unspent transaction outputs* and every bitcoin account holds a set of

utxos which are just quantities of bitcoin that have been sent to that account that have not been redeemed yet and these utxos can contain any quantity of bitcoin but once they're spent you have to spend everything that's inside of it .so I think it's really helpful to think of these as piggy banks so let's say I have a piggy bank that has three bitcoin in it and if you want to access what's in that piggy bank you have to break it open you just have to destroy the entire piggy bank and so once it's destroyed you can't put it back together again and use it it's destroyed it's gone forever so it's the same thing with bitcoin. so now I'm holding three bitcoin in my hand and I want to pay one of those to someone I want to pay point five of that to my tutor let's say and I want to keep 1.5 for myself but I had to break open the whole piggy bank to use that right now I had to break open the whole three bitcoin so what do I do with that 1.5 that's left over that I want for myself I can actually reroute that and I can send it back to myself but I just created three new piggy banks.so I just sent one piggy bank to someone and that's now his utxo I've sent 0.5 to my tutor and I've sent 1.5 back to myself and so this kind of illustrates the idea that it has to be spent in its entirety and it can also only be redeemed once so the second you open up that utxo it's like that utxo's gone like you need to create new utxos now and you can send you can send it and you can spend those utxos you can send it to new people or you can send it back to yourself as in picture below we have Alice with two utxos one with one bitcoin and the other with 1.2 bitcoins here Alice wants to send two bitcoins to Bob so she breaks her all piggy banks(utxos) sends two bitcoins to Bob in a new piggy bank and sends 0.2 bitcoin to herself in another new piggy bank.



Now let's see the validity we talked about in this model:

1. proof of ownership:

Signature generated by private key (it is like thumb print when you thumbed a transaction you are proving that you own those piggy banks)

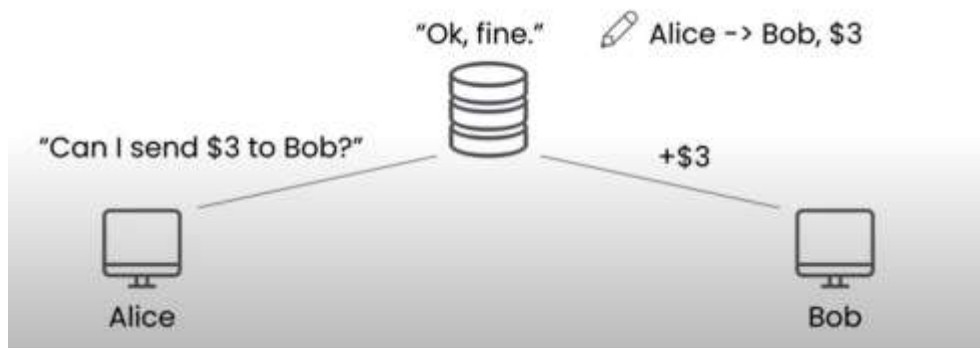
2. Available funds:

UTXOs are inputted directly.(because those UTXOs go directly into transactions if the left and right side of the transaction won't be equal the transaction won't happen)

3.No other transactions using the same fund(we'll get to that later)

Distributed Ledger:

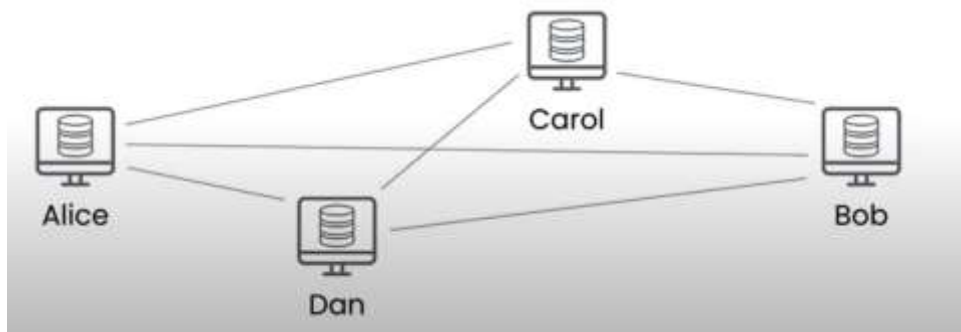
Just like the last part first we are going to describe the traditional model for record-keeping and then we will talk about bitcoin.so in traditional record-keeping data is stored and updated by one central party(like bank) and it works somehow like the picture below:



And the bank should have security measures in place to prevent hackers and failure.

So as we said earlier trusting that third party is an issue it self so blockchain kind of removes that third party and does something like that:

In blockchain data is stored by and updates are broadcasted to everyone.(so everybody gets the data and everybody stores it and everybody decides if they want to update) so it is transparent and fault-tolerant by design (because everyone get to access the data and store it if someone messes with the data it is noticeable) something like the picture below:



Now you might be wondering how exactly are these transactions stored on the blockchain and you may think like each individual transaction becomes like its own block or maybe it gets stored by itself but what actually happens is if that were to be the case every single person interacting with that blockchain would have to update every single time a transaction occurred and so you can imagine this is super time consuming and super repetitive so what ends up happening is

that we bundle these transactions into blocks and that's where the block in blockchain comes from with backlinks to enforce an ordering so each block references the block before it and the block before it and the block before it and this is just to ensure that we have valid transactions happening that we're operating on the most up-to-date.(the transactions get broadcast to everyone in the network and then once somebody has a new block that they want to add to the blockchain they say okay I'll just grab these transactions that were just sent to me we'll bundle them all together and we'll make sure that they're valid and then once we add them to into the block then that person will broadcast that new block to everyone else and say hey look at this I found a block that contains these transactions in it and that's kind of how the blockchain gets added onto is that these people continue to find blocks and then every time someone finds a block they put new transactions into) **Trustless**

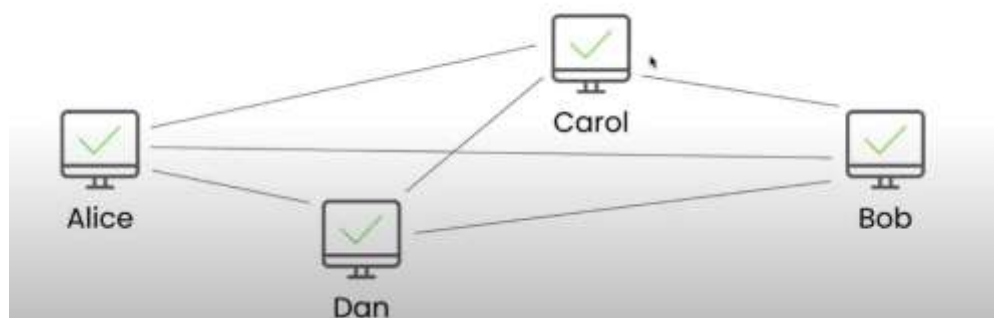
consensus:

Consensus: is the process by which a network's participants come to agreement on some decision to be made(in our case agreeing on changes to a ledger of transactions).

Traditionally we place 100% trust in bank.

There are different types of consensus:

1.Naive consensus: everyone accepts valid transactions as they come around without discussion.



One problem that you can probably guess is the idea of a double spend attack and what this means is if Alice has like one

bitcoin and she wants to buy an iphone from carol and a mac from bob with naive consensus if everyone just agrees every transaction that happens then I can take my one bitcoin or Alice to take her one bitcoin and send it to carol but then at the same time take that same one bitcoin to bob and so if we think with like fiat currencies like the dollar or euro we can't really do this like if I go to a store and give someone five dollars for a candy bar or whatever it is I can't get those five dollars back I have a candy bar and I've spent that five dollars but with bitcoin with naive consensus if we want to spend a currency once and then reuse that currency just because we can like copy and paste it essentially we can find that through naïve consensus it's really challenging to detect when a certain currency has been spent multiple times and this like artificially inflates the value of bitcoin.

2.Democratic consensus: instead, let's have proposers that broadcast transactions, and votes that choose whether or not to include them.

so now if we try the same attack that we tried earlier we have Alice trying to spend the same bitcoin multiple times with multiple different people let's if everyone interacts with each other and everyone votes democratically on consensus to agree with like how the transactions are going then we see that because carol and bob and dan are actually paying attention and they're interacting with each other and comparing their own transactions we see that well Alice only has one bitcoin Alice doesn't have two bitcoins and so because of that just like in the same way that we can't go into debt or spend more bitcoin than we have everyone sees that Alice only has one bitcoin and so they disagree completely they just say that this transaction is not valid and by invalidating it then we don't add to the network and Alice can't get away with this double

spending problem but the problem is because it's so simple to make multiple identities and have different accounts Alice can just make a bunch of different versions of herself and so you can have Alice two three and four so now because of all the different computers and devices if it's a purely democratic system then Alice actually is taking control over the network because in terms of voting Alice has four votes and everyone else has three and so Alice can just okay all the different transactions on this network and really just have like an artificially and like into infinite like an infinitely inflated amount of bitcoin to interact with)

3.Nakamoto consensus: Now, let's make voters do a bunch of pointless brute-force computation to be able to cast a vote.(if Alice has a lot of devices because the task that you need to do to get the vote is really difficult she cannot do that for all her devices)

So after all if you remember at the first pages we pointed out some qualities for a currency now we are going to check if we can say that bitcoin is a currency or not.

- 1.durability: persisted forever on the blockchain.
- 2.portability: can keep your whole balance in your pocket.
- 3.divisibility: exchangeable in arbitrary quantities as little as 0.00000001 BTC.
- 4.uniformity:no distinguishing characteristics that would make one bitcoin worth more than another.
- 5.limited supply: finite cap of 21000000 BTC.
- 6.Accebility: particularly acceptable because you don't have to place trust in anyone else in order to redeem your money.

So to sum it up you can say that bitcoin is a currency.

Now let's summarize what we learned today:

Identity: you use your public key to receive bitcoin, and your private key to redeem it.

Transactions: you own a set of UTXOs that you can input into transactions.

Distributed ledger: Each party is responsible for maintaining a copy of the blockchain.

Trustless consensus: Transactions are approved via proof-ofwork, an expensive voting process, to deter double spend attacks.

Lecture 3: Bitcoin mechanics and optimization

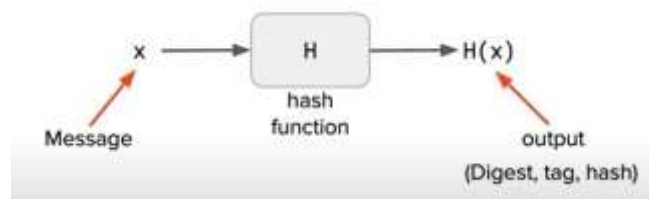
<https://www.youtube.com/watch?v=Z-peRVXllw0&list=PLLkiRvMHnp6OIWkZVa85g2tv7NtlgoAID&index=3>

Cryptographic hash functions:

So we did talk about identities and its types and so ever now we are going to talk about how blockchain and bitcoins identify people. pretty much everything in blockchain relates to identities there isn't like a username password kind of thing in blockchain so the way you would verify that someone exists it's very challenging to do because there is no concept of social security numbers and license or licensed driver's license or anything in blockchain so the fundamental question starts that how can we trust someone in a trustless environment how can we trust that this person exists or am I sending money to the correct person when there isn't like a formal proof that this person exists or like not a formal proof as in like an identifications so this is the beginning idea of what hash functions cryptographic cache functions usually originate with

and later on down the line was kind of used well in bitcoin using digital signatures but this has been around for a very long time as well.

So you can refer to hash function as a function in math you give some certain inputs to the function and then it does some procedure on that input and it gives you the output. In hash function the inputs will be messages and the outputs will be pictures and what's special about this is that you can go from x to h of x pretty easily and you can convert any messages to a digest pretty easily but it's extremely challenging and extremely difficult to go backwards so you cannot generate an x from an h of x even though it was like the same x that generated h of x .



And another amazing feature is that no matter what size of message or information you give to the hash function the output will always have the 256 bits size.

So before we introduce the hash function you might be wondering can any function be a cryptographic hash function?

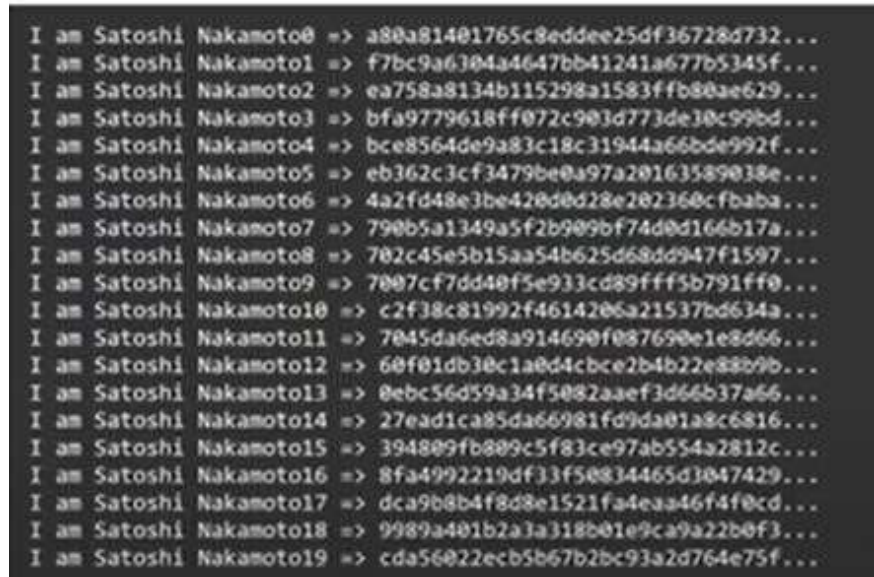
A hash function has three properties:

1. Computationally efficient (Set of computation to get a digest/hash should not take a long time in other words it shouldn't take much time to generate h of x out of x but remember it shouldn't take very little time to. Because if the tasks in the function are easy and it literally doesn't take any time means that you can generate x out of h of x)
2. Collision resistance (it should be hard to find two inputs that maps the same output/hash/digest. the output should look

random just like finger prints you can not find two people with the same finger print)

3.Hide information (given the output, it should be hard to find anything interesting (like x is an odd or even number) about the input it kind of leads to the meaning that if your inputs have very little differences the outputs should differ which causes the avalanche effect) **Avalanche effect:**

The input produces a pseudorandom change in the output.as you can see in the picture below the inputs are almost the same but at the end of them they have different digits and the outputs are really different from one another.



```
I am Satoshi Nakamoto0 => a80a81401765c8eddee25df36728d732...
I am Satoshi Nakamoto1 => f7bc9a6304a4647bb41241a677b5345f...
I am Satoshi Nakamoto2 => ea758a8134b115298a1583ffb80ae629...
I am Satoshi Nakamoto3 => bfa9779618ff072c903d773de30c99bd...
I am Satoshi Nakamoto4 => bce8564de9a83c18c31944a66bde992f...
I am Satoshi Nakamoto5 => eb362c3cf3479be0a97a20163589038e...
I am Satoshi Nakamoto6 => 4a2fd48e3be428d0d28e202360cfbaba...
I am Satoshi Nakamoto7 => 790b5a1349a5f2b909bf74d0d166b17a...
I am Satoshi Nakamoto8 => 702c45e5b15aa54b625d68dd947f1597...
I am Satoshi Nakamoto9 => 7007cf7dd40f5e933cd89fff5b791ff0...
I am Satoshi Nakamoto10 => c2f38c81992f4614206a21537bd634a...
I am Satoshi Nakamoto11 => 7045da6ed8a914690f087690e1e8d66...
I am Satoshi Nakamoto12 => 60f01db30c1a0d4cbce2b4b22e88b9b...
I am Satoshi Nakamoto13 => 0ebc56d59a34f5082aaef3d66b37a66...
I am Satoshi Nakamoto14 => 27ead1ca85da66981fd9da01a8c6816...
I am Satoshi Nakamoto15 => 394809fb809c5f83ce97ab554a2812c...
I am Satoshi Nakamoto16 => 8fa4992219df33f50834465d3047429...
I am Satoshi Nakamoto17 => dca9b8b4f8d8e1521fa4eaa46f4f0cd...
I am Satoshi Nakamoto18 => 9989a401b2a3a318b01e9ca9a22b0f3...
I am Satoshi Nakamoto19 => cda56022ecb5b67b2bc93a2d764e75f...
```

Now let's tell you that bitcoin uses the SHA-256² hash function($H(x)=\text{SHA256}(\text{SHA256}(x))$) which is built by NSA.

A temper evident database: when we talk about a blockchain we talk about it as a tamperevident database what does this mean? a tamper-evident database in the context of blockchain is kind of like a appendonly ledger that someone is able to change the data on a given block or on the blockchain in general it must be able to be detected really easily by the

nature of the mining protocol as well as the proof of work you will ensure that this block does not get included on the blockchain forever since basically miners will no longer include it or fork their own blockchain and start over.

So into the each block of blockchain we have these things:

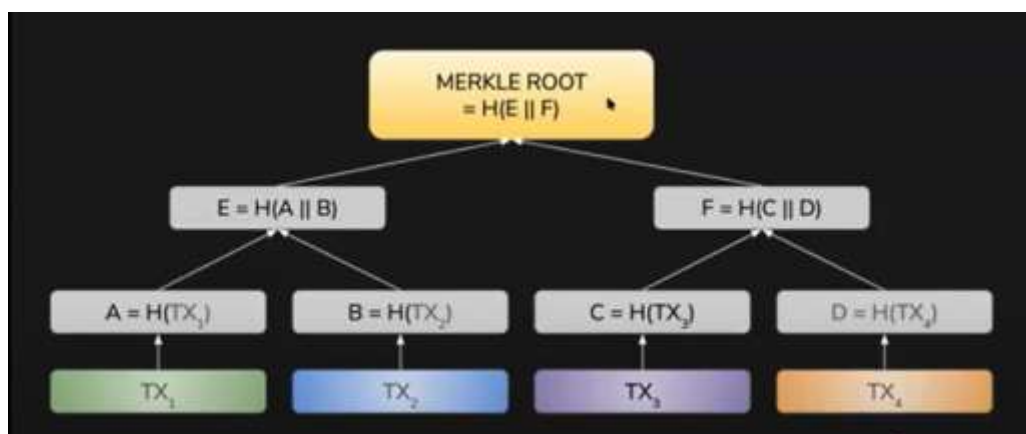


within each block we have kind of these three fields that define the block header and the block header will construct what is known as a block ID the block ID is simply just like a unique identifier for each block you can tell that a given block is blocked by its block ID and it contains three main components.

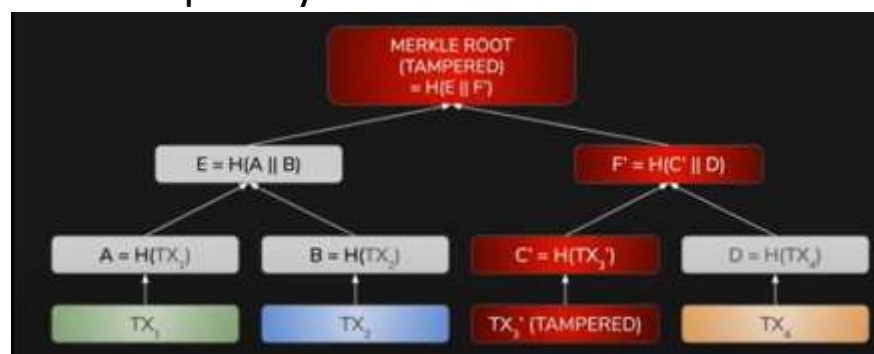
$\text{blockID} = H(\text{blockHeader}) = H(\text{prevBlockHash} || \text{merkleRoot} || \text{nonce})$

Now let's discuss every component.

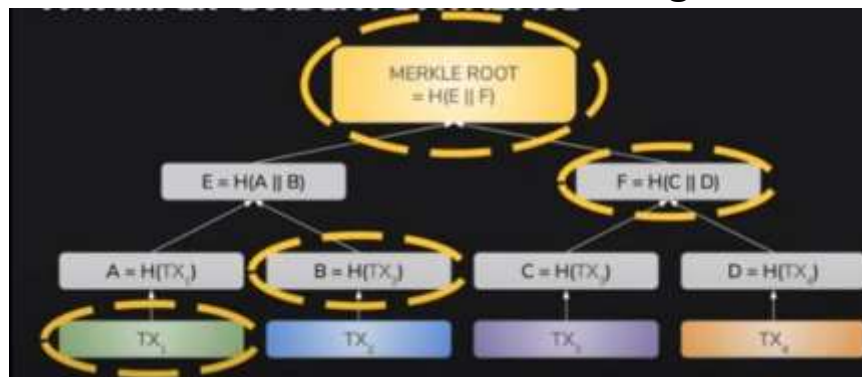
1. Merkle root (hash binary tree): it's a data structure that allows for an efficient representation of transactions as well as a quick validation of existence within a given tree. so this is a merkle root:



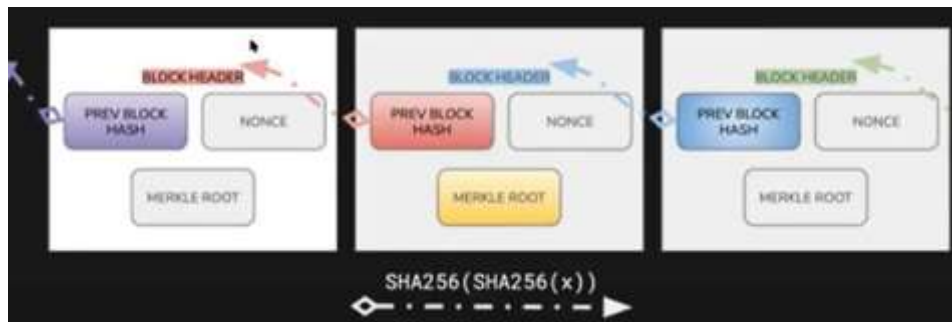
how it's constructed for those of you with a cs background you might recognize this as some sort of binary tree. you have your root node and stemming from each root node you have one or more children nodes and as you can see here we have the leaf nodes(the leaf nodes represent nodes that are at the bottom of the tree meaning that they don't have any children) and basically all your end transactions will be the leaf nodes and at each level above the leaf nodes you will be basically taking a hash of all the children concatenated together so if we see at the second from the bottom level you will see that A equals of transaction x so this is some 256 bit string that is a unique representation of transaction one and is given the transaction a hash of transaction one and as we go up the chain we hash the two children together and eventually we get to our merkle root, our merkle root is basically kind of in some sense a hash of all the previous transactions but there is some recursion involved. the reason why a merkle root is so powerful is because just by storing a merkle root any changes to the transactions themselves will propagate through the tree all the way up to the merkle root so in the case that transaction three has been tampered with is c if we see from before was the regular c will be changed to c prime since the hash will change completely and then f will change to f prime and then our vertical root is completely different now.



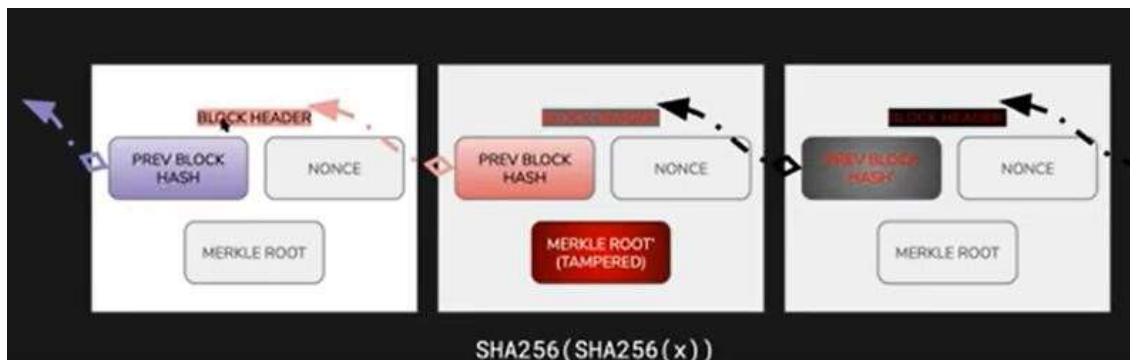
in the case that someone wants to verify that a list of transactions are all included in a given block they will compute the merkle root and eventually end up with the merkle root and if the merkle root that they compute is different from the merkle root that's on the block then something has been tampered in which case they should not accept this block. another powerful aspect of a merkle root is that it requires $O(n)$ nodes that will need to be returned in order to determine whether a transaction is included in the block or not. For example if I say I want to ensure that transaction one is in a given block instead of passing transaction one transaction two transaction three transaction four and so on and computing all of their hashes and eventually computing their merkle root I just need to give you transaction one the hash of transaction 2 and the hash of the hash of transaction 3 concatenated with the hash of transaction 4 and if you kind of see here it's the highest node that is not along the path from the transaction to the root so given this I can compute the merkle root and compare it to the merkle root that's in the block header and check whether a transaction is included in a given block.



2. Prev Block Hash: next component of block header is your previous block hash a previous block hash basically creates an implicit link between a given block and previous blocks maintaining the chronological order of the blockchain and also ensuring that a previous block has not been tampered with.



$\text{prevBlockHash} = H(\text{prevBlockHash} || \text{merkleRoot} || \text{nonce})$
 if you look at this diagram you have a given block that has a hash of the previous block and the previous block hash through. like the power of the recursive property of a blockchain you're able to basically capture the integrity and in nature of previous blocks through the previous block hash because when you hash a previous block you're also hashing the previous blocks hash in the block header and so you're hashing the previous block as well and that recursion kind of holds through all the way to the start of the blockchain. in the case that a merkle root has actually been tampered with meaning that a different transaction has been included than that which was proposed in the block in the given block you will have the previous block hash will be different since the previous block hash is a hash of the previous block hash concatenated with the merkle root and the nonce.



3. Nonce

Nonce is basically a number more formally it's a bit string that we use to generate randomness for a block header to solve what is called a hash puzzle.

Before getting to the rest about the Nonce we need to ensure that we know about proof of work consensus. when we talk about proof-of-work consensus and consensus protocols in general it's basically a way for minors to spend some sort of resource(in this case proof of work means they're spending some time as well as computational power)that they're putting the work in to prove that a given block is valid and they basically mined it and in the case of bitcoin this proof of work puzzle that one does is that they repeatedly compute a hash of the block header which includes the hash of the previous block the merkle root as well as the Nonce and they have to ensure that this hash actually is less than a given target(to sum it up Bitcoin's proof of work consensus requires miners to solve a computationally difficult puzzle) So the hash puzzle needs to have some properties:

- 1.it should be computationally difficult (which means it can't be too easy otherwise an individual who has a lot of hash power (which is directly proportional to the how efficient your hardware is) will basically have control over the blockchain because they can publish blocks faster than anyone else)
- 2.it should be adjustable (it should be able to change as mining power increases)
- 3.it should be easily verifiable (so once a block has been published other individuals (other miners) can say okay I verify this block by computing the hash of the block header and seeing that it's less than the target and now I'm going to add it onto my personal blockchain)

In the case of bitcoin we have something that is called the partial pre-image hash puzzle is the problem of finding a nonce that satisfies the following inequality that the hash of the previous block, hash concatenated with the merkle root and the Nonce is less than target (

$$H(\text{prevBlockHash} \parallel \text{merkleRoot} \parallel \text{nonce}) < \text{target}$$

For solving this equation we know that inverting a hash is very difficult, in this case we're trying to compute a partial preimage hash we're not actually trying to invert the hash but we're trying to create a hash that outputs some value that is less than a target so it's really random the best way to kind of represent this is by throwing darts at a target while blindfolded in reality since there's no rhyme or reason to a hash function output so there's no correlation between different inputs and different outputs and you have an equal likelihood of hitting any part of the target. the faster throw words can correspond to more hits per second the analogy between faster throws on a dartboard is faster miners or more powerful miners in the mining community and so miners will throw these random darts which is literally just computing hashes of their block header and incrementing the nonce and they will check that it's below the target if not they will increment the nonce and Continue.

Now let's show an example:

```
H(prevBlockHash || merkleRoot || nonce) < 0x0000
```

We assume that the green block is our target. If we put the nonce=0 we have:

```
0x1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64  
0x0000ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
```

As you can see with nonce=0 the equation won't be true then we try nonce=1 we have:

```
0xe9afc424b79e4f6ab42d99c81156d3a17228d6e1ee f4139be78e948a9332a7d8
```

As you can see it is not a good answer either so if we continue and try nonce=4250 we have:

```
0x0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e
<
0x0000ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
Solved!
```

We can see that this is a valid hash and we've solved this partial pre-image hash puzzle so now we include it in our block header and publish the block for other miners to add it to their blockchain. when we talk about a target, a target is a function of the difficulty as well as time it takes to hash the previous 2016 blocks. we represent difficulty more formally as the expected number of computations required to find a block as a requirement of leading numbers of zeros so this is a tunable value that we change based how this partial pre-image hash puzzle must be you must be able to change its difficulty and so that we are able to tune this difficulty based on the global hash rate and this adjusts every 2016 blocks and if we assume that mining a block takes about 10 minutes then this will take about two weeks. For example:

```
H(prevBlockHash || merkleRoot || nonce) <
a. 0x0000ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
b. 0x0000000000000000000000000000000000000000000000000000000000000000
c. 0x00000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
```

So a is the easiest and b will be the hardest.
Now this is a code to find nonce:


```

TARGET = (65535 << 208) / DIFFICULTY;
coinbase_nonce = 0;
while (1) {
    header = makeBlockHeader(transactions, coinbase_nonce);
    for (header_nonce = 0; header_nonce < (1 << 32); header_nonce++){
        if (SHA256(SHA256(makeBlock(header, header_nonce))) <
            TARGET)
            break; //block found!
    }
    coinbase_nonce++;
}

```

Figure 5.6: CPU mining pseudocode.

SIGS, ECDSA, AND ADDRESSES

Dilemma: when sending transactions to other users, we want two seemingly contradictory things to happen:

1. Tie user identity to a transaction
2. Have no sensitive identifiable characteristics associated with a particular transaction.

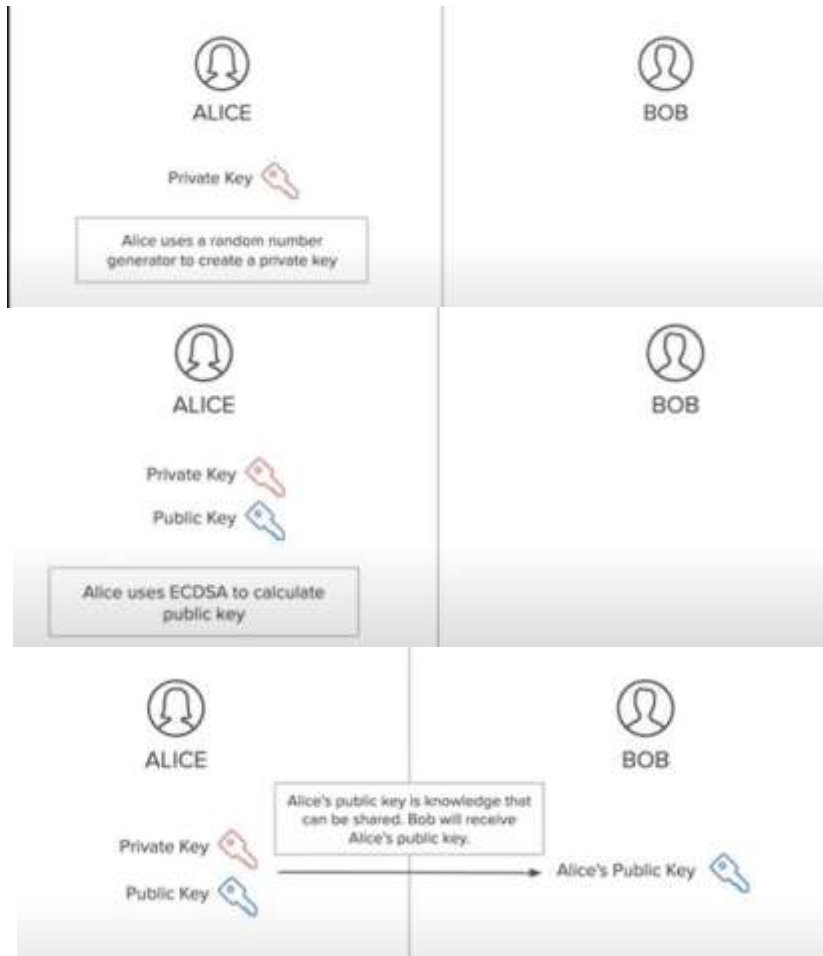
So the way we solve this dilemma is:

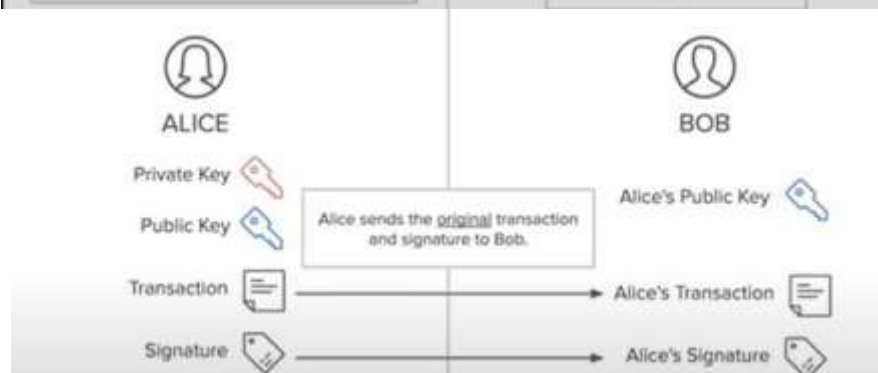
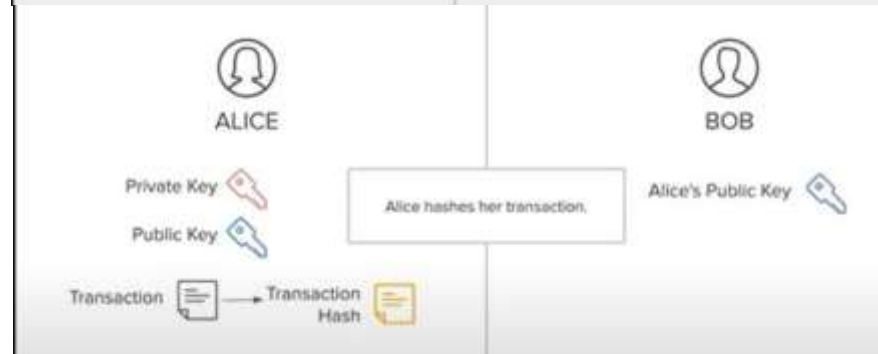
Public key cryptography: a cryptographic system that allows for secure dissemination of identify and authentication of valid messages (public key cryptography is a scheme which allows an individual to prove that they are who they are and also attach that proof to a given message without revealing too much about themselves or any kind of token or identity that would allow their identity to be compromised)

Actually we have two keys that form the public key cryptography:

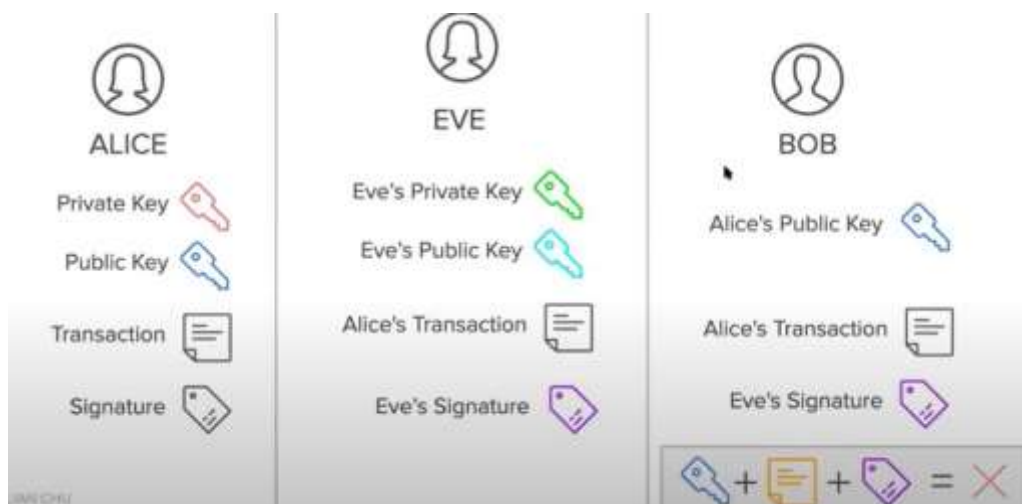
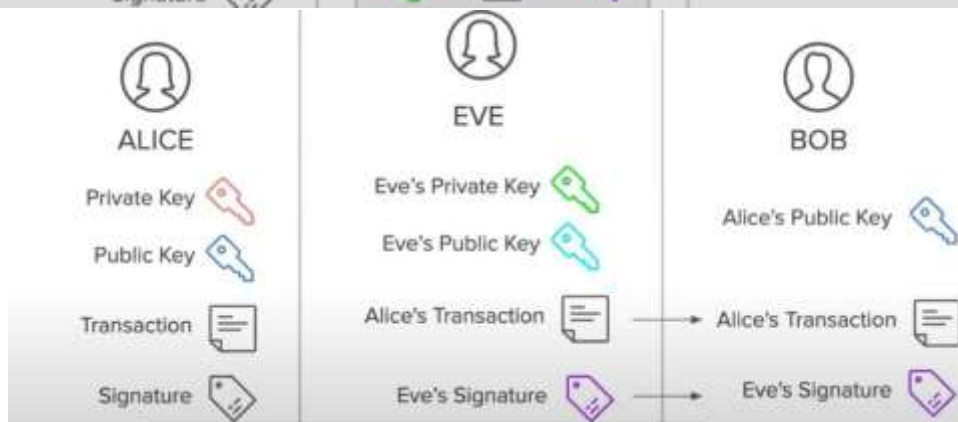
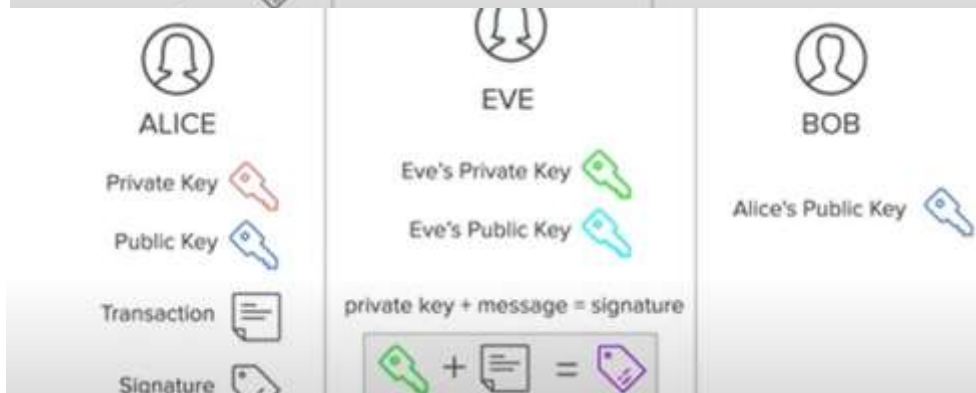
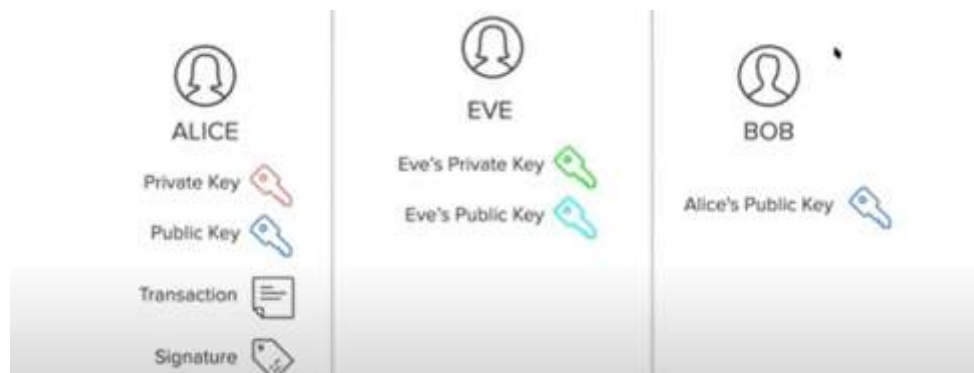
1. public key: information about a user that can be distributed widely.
2. private key: sensitive information about a user that should be only known by the user (if someone had access to your private key they would be able to compromise your account so the secrecy of the private key is super important). the scheme that we talk about for public key cryptography as well as the digital signature algorithm is ECDSA which is based on elliptic

curves(which is a mathematical construction). Now let's hear an example about Alice and Bob trying to do a transaction.





Now if some one wants to temper with the transaction for example eve. We have:

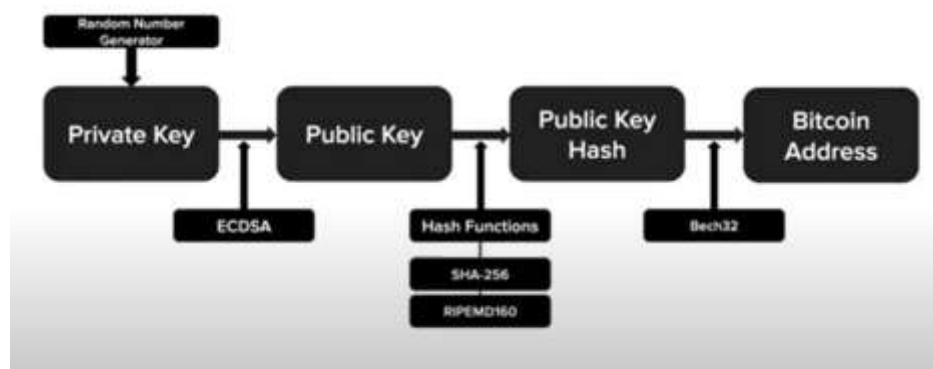


Now let's summarize everything we said:

Recipients given the (message, signature) pair should be able to verify:

1. Message origin: original sender (owner of private key) has authorized this message/transaction
2. Non-repudiation: original sender (owner of private key) can't backtrack
3. Message Integrity: transaction cannot have been modified since sending.

If you wanna address your bitcoin here is what you should do:

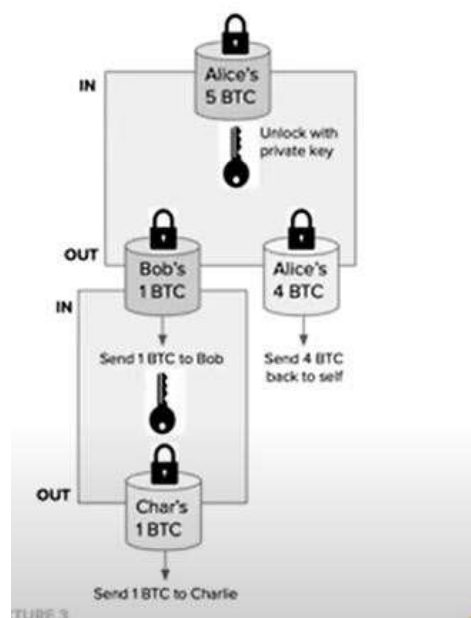


SHA-256 is a secure hashing algorithm and is used extensively in bitcoin scripts and mining.

RIPEMD is the RACE integrity primitives evaluation message digest and it produces 160 bit number. *Bitcoin script:*

so far we've gone over cryptographic hash functions those are just like building blocks of using cryptographic hash functions to generate public keys and making the system more secure and then we talked about tamper evident database with merkle trees and stuff and then we talked about six signatures ECDSAS and bitcoin addresses, all of that was to generate your identity in a way that it's secure and it's like transmitted across the network efficiently now we're going to talk about scripts which

is a way that you can send transactions between two parties where you don't know anything about them so it's like you can send transaction right now between wallets using visa because you know the social security address of someone else or you know their bank account and you know their routing number account and you can send the transactions but in bitcoin's case you don't know anything about this person this person is unknown to you so how can you verify that this person is actually the correct person that you're sending the information to and that is going to be extremely pivotal in the next step which is going to be using bitcoin script. bitcoin uses a programming language called scripts and that is primarily used to generate the locking and unlocking scripts we'll talk about that in a bit but if you remember about the utxo models where you have inputs and outputs. the locking script is called the output script and the input script is called the unlocking script so you're going to have a certain set of outputs that you're going to send out and you're going to lock it up with a key called script pub key and then the inputs are being locked by script sig so whenever you have to access the inputs you'd use your script sig and then actually access those

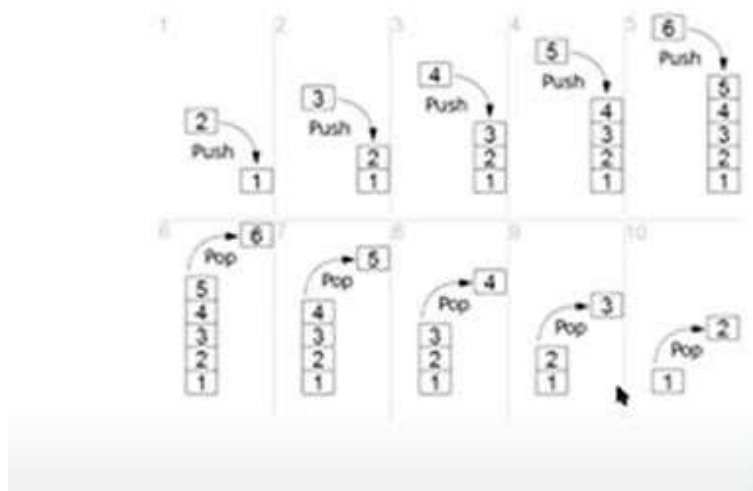


script programming language

some of its properties are: -No loops(not turning complete) - stack based

-several operators

So it doesn't have for loops or while loops or do whiles or anything like that it's just a basic set of one-line commands one not even one line it's just like one block commands and then it's stack if you check the picture below:



you're pushing some elements up and making like a tower and then at the end you're just like popping the topmost element out of the stack so it goes back to the beginning one so that is what a stack based programming is and it just like stack bunch of things and then get rid of them from top to bottom so that is the programming language that bitcoin uses to just generate scripts of course bitcoin is written in like the script is actually written in c plus plus the test is written in python but this is just to generate scripts and one of the most important examples is where you can see it in p2pkh which is widely known as *pay to pub key hash*

```
P2pkh: OP_DUP OP_HASH160 <PKHash> OP_EQUALVERIFY OP_CHECKSIG
```

this long line of command that you see what we call a script in generating the script pub key and this is in fact the pub key a

way to verify the pub key in bitcoin a more simpler example of what a script based language actually can generate is something like the picture below:

Simpler example: OP_3 OP_4 OP_EQUAL OP_IF OP_5 OP_ELSE OP_10 OP_ENDIF

if we actually follow this through we will get something like op3 means that well there's an operator called three so we'll just add in three in the stack op4 will add four in the stack and then we have three and four in the stack and now op equal checks if those equal to five and if it equals then print five all or else if it doesn't equal then print ten ultimately 3 does not equal 4 so we are printing out 10 at the end. these are more commands of bitcoin scripts:

1. OP_TRUE
2. OP_FALSE
3. OP_NOT
4. OP_NOP
5. OP_RETURN <20 bytes in hex>
6. OP_ADD OP_SUB OP_2 OP_EQUAL
7. OP_HASH256 <32 bytes in hex> OP_EQUAL
8. 2 <pubkey A> <pubkey B> <pubkey C> 3 OP_CHECKMULTISIG

this is the p2pkh in action:

SCRIPT: <Signature> <PublicKey> OP_DUP OP_HASH160 <PKHash> OP_EQUALVERIFY
OP_CHECKSIG



if you see top right hand corner this is like a script sig is present in the input of the transactions right here and then script pub key is present in the output of the transactions so to verify that

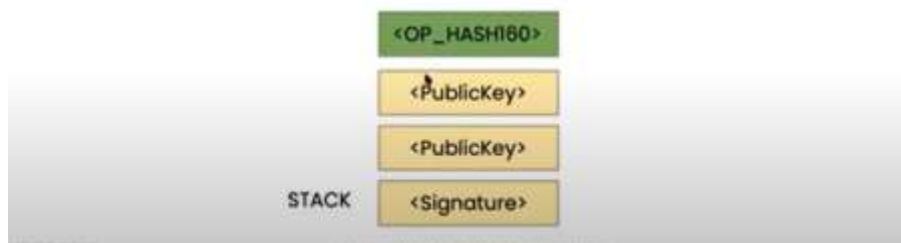
this transaction is sent from the correct user and then received from the correct user we would actually concatenate both of them so the first two term right here signature and public key is from the input and then the rest is from the output trend the output version of the transaction we're just going to have signature in the stack at the start then we're going to drop it in and then next thing we're going to drop in pub key of course

```
SCRIPT: <Signature> <PublicKey> OP_DUP OP_HASH160 <PKHash> OP_EQUALVERIFY  
OP_CHECKSIG
```



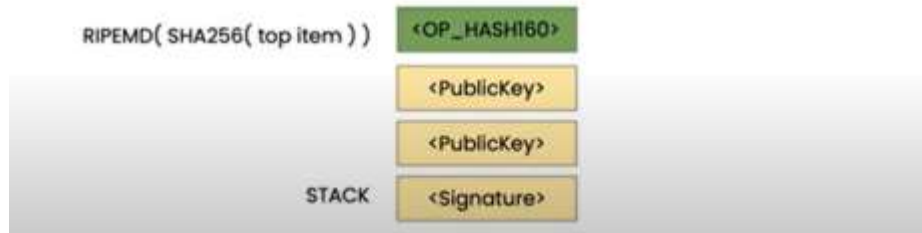
and those are our script sigs and then the next thing we're going to do is OP_DUP OP_DUP basically means we're going to duplicate the um the topmost element in our stack so it will just duplicate it so now we have two pub keys

```
SCRIPT: <Signature> <PublicKey> OP_DUP OP_HASH160 <PKHash> OP_EQUALVERIFY  
OP_CHECKSIG
```



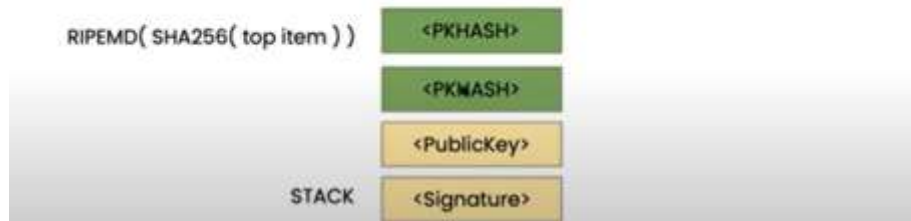
and then the next thing is OP_HASH160 is basically saying to use like ripemd hash in like the topmost level of our stack so this it's going to hash the public key that is at the topmost level

SCRIPT: <Signature> <PublicKey> OP_DUP OP_HASH160 <PKHash> OP_EQUALVERIFY
OP_CHECKSIG

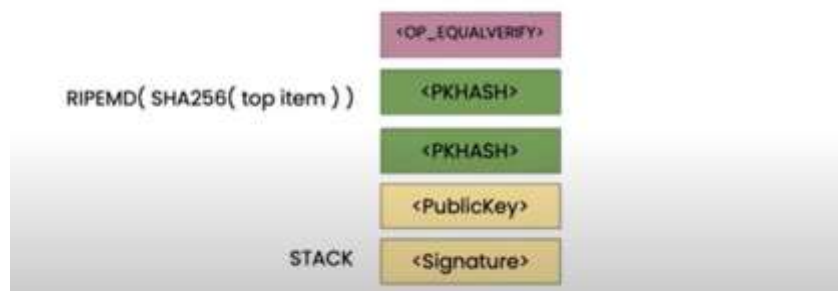


the next up we do is we input the public key hash into the stack as well

SCRIPT: <Signature> <PublicKey> OP_DUP OP_HASH160 <PKHash> OP_EQUALVERIFY
OP_CHECKSIG



so now the next thing that we run is OP_EQUALVERIFY
it just checks whether the top two elements are equal and then
if in any of these cases if the output is false and the boolean
value is false the process just terminates then you can't make
the transaction at all so uh whatever you check and verify
you're basically checking whether those public key hashes
matches or not



so these two hashes has to match up and when it matches up
that's a good sign so we move on to OP_CHECKSIG and it does
it checks whether um the public key can be derived with that
signature um because if it's not then it's definitely not a good

probably key and it's definitely not a valid public key so the transactions wouldn't be a valid transaction

SCRIPT: <Signature> <PublicKey> OP_DUP OP_HASH160 <PKHash> OP_EQUALVERIFY
OP_CHECKSIG



so when OP_CHECKSIG does a final check on the transaction then if everything is true the stack returns true or one and then the user is good to go and they can make a transaction

SCRIPT: <Signature> <PublicKey> OP_DUP OP_HASH160 <PKHash> OP_EQUALVERIFY
OP_CHECKSIG



so that is the final level.

Lecture 4: Interacting with bitcoin

https://www.youtube.com/watch?v=6JV4dGX_ABU&list=PLLkiRvMHnp6OIWkZVa85g2tv7NtlgoAID&index=4

Types of users

not every client is a miner because most people don't have the space or the money to buy a powerful computer to mine blockchain and not everyone wants to spend the time mining blockchain and not every client could download the entire blockchain because many devices don't have enough storage space to download the entire blockchain for example if you want to send bitcoin with your phone it'll be impractical for you

to download the entire blockchain because most phones don't have 280 gigs of memory and not every client is directly connected to the network because if you're not making regular transactions then you don't need to communicate with a network daily which could take up a lot of bandwidth, and not every client has a wallet because they could be using another service as their wallet.

A node is a computer connected to other computers which follows rules and shares information. A 'full node' is a computer in Bitcoin's peer-to-peer network which hosts and synchronises a copy of the entire Bitcoin blockchain. Nodes are essential for keeping a cryptocurrency network running. full nodes are usually what's used by miners and people who are more connected to blockchain but most people would not need a full node. **Light Nodes(SPV):**

Simple payment verification is method for verifying if particular transactions are included in a block without downloading the entire block.

Wallets

What is a Bitcoin wallet?

On a practical level, a Bitcoin wallet is a device or program that is used to send and receive bitcoins. The term **wallet** can be a little confusing for people new to Bitcoin and crypto. A physical wallet is used to store physical currency, however a Bitcoin wallet does not store bitcoins within it. How can this work? Well, most people already have something similar to this in their physical wallets right now: a debit card. The debit card in your wallet is not money, but it does grant you **access** to your money. This is similar to how a Bitcoin wallet works, with a key difference being that while a debit card is controlled by a

centralized entity (a bank), no person or organization controls Bitcoin. This difference means that Bitcoin wallets must function somewhat differently than bank accounts.

How Bitcoin wallets work

Continuing with the debit card analogy from above, a Bitcoin wallet holds at least one "account," or sub-wallet. We can view this sub-wallet as being roughly equivalent to a debit card. For instance, debit cards have information associated with them, including an account number and a password. Each Bitcoin "account" within a Bitcoin wallet also has information associated with it. For our purposes, the two key pieces of information are the public bitcoin address and the private key. The public address is comparable to the debit card's account number. The private key, meanwhile, is kind of like a debit card's password in that it grants access to the bitcoin associated with that public bitcoin address. A private key is a 256-bit secret number. Here is an example:

```
108165236279178312660610114131826512483935470542850  
824183737259708197206310322
```

As you can see, this secret number is a bit unwieldy. One of the key functions of a Bitcoin wallet is to manage the private key. In fact, private keys are almost never handled directly by users. Bitcoin wallets provide a way to write down this private key in a much more human readable format, referred to as a recovery phrase, secret passphrase, or seed phrase. A recovery phrase is a list of words, usually between 12 and 24, that allow you to reconstruct your Bitcoin wallet and gain access to your funds even if your Bitcoin wallet is destroyed. Here is an example of a recovery phrase consisting of 12 words:

While the recovery phrase is an improvement upon the private key, it still leaves a lot to be desired. Since you shouldn't store your recovery phrase in plain text (unencrypted form) on your

computer, for most people the best solution is to write it down on paper. This presents problems because safely storing a piece of paper can be hard. Further, if you're using a multi-coin wallet (like the **Bitcoin.com Wallet**), you'll have a separate recovery phrase for every different blockchain your wallet supports. Storing all those recovery phrases on paper quickly becomes onerous. For this reason, the Bitcoin.com Wallet integrates a "Cloud Backup" system. Here you can create a single custom password and use it to unlock all of your private keys, which are stored in encrypted form in your Google or iCloud account. to secure our identity we need to secure a private key which is kind of like our password for example you don't want people getting your facebook your email passwords because there'll be a lot of private information on there this is even more important for blockchain because it's money that you're trying to secure so how do we manage all of our private keys? we'd use a wallet.

A blockchain wallet is a cryptocurrency wallet that allows users to manage different kinds of cryptocurrencies—for example, Bitcoin or Ethereum. A blockchain wallet helps someone exchange funds easily. Transactions are secure, as they are cryptographically signed.

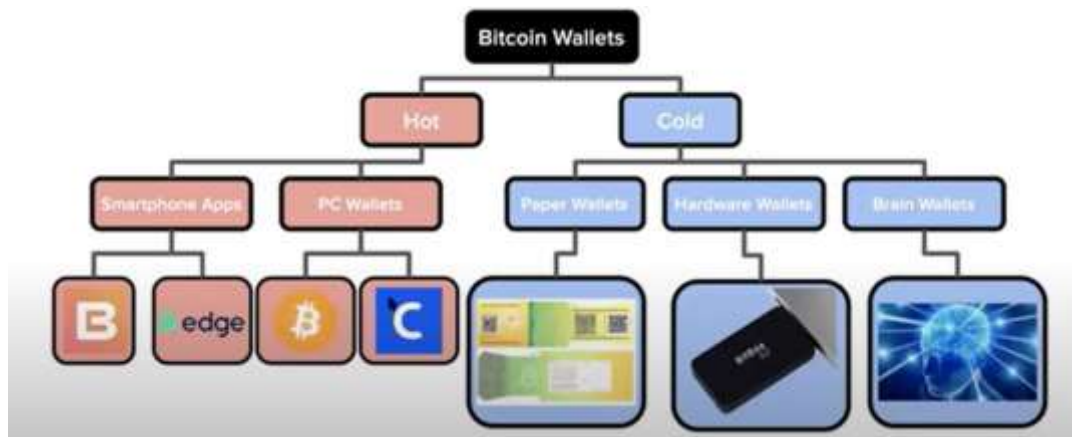
What do wallets do?

- provides a user interface to the blockchain.
- keep track of your private key
- store, send, receive, and list transaction.

Types of wallet:

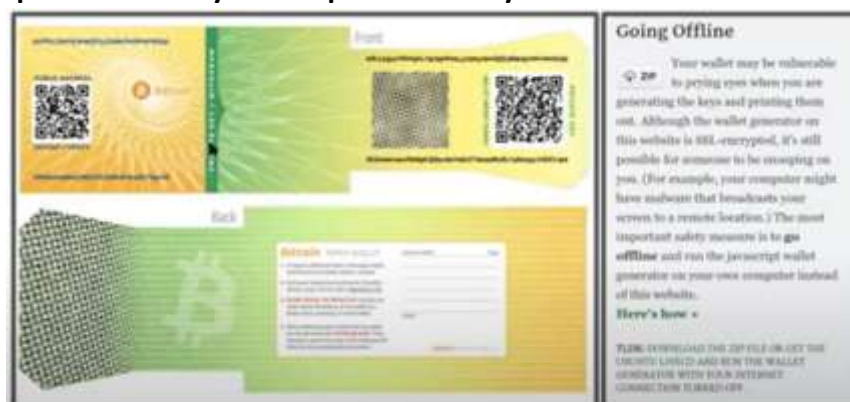
1.Hot wallet (hot wallets are hot because they're reliant on the internet which is another layer of vulnerability because it's easier to break into a wallet if you are connected to the internet).

2. Cold wallet (cold wallets are not connected to the internet which is better because they're more secure but then they're harder to use and they're usually much simpler too).



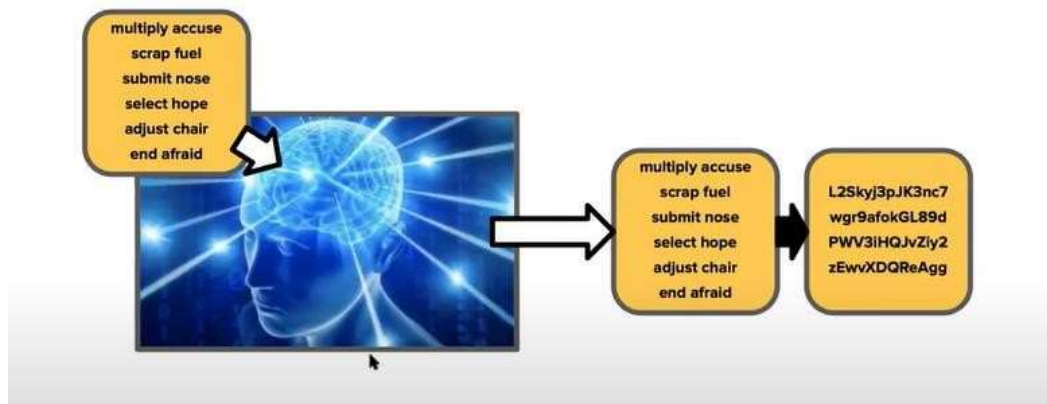
Cold storage wallet: a device that's disconnected from your computer you can use it to store your private key and you also use it to sign transactions there might be some other features based on how fancy it is but it's mostly just a device to keep your private key safe.

Paper wallet: writing down your private key and public key you can scan it with a QR code and yeah if you have a paper wallet you need to keep it safe so because if people find it then they know your private key and public key.



Brain wallets: it's pretty much you're memorizing your public key and private key and a good way of doing that is having a mnemonic, or collection of words/phrases so then you can use the first letter of each word to generate your private key it's easier to have something that you could memorize because

memorizing a sequence of letters and numbers is pretty hard to do but this may not be very secure because humans aren't as random as we think we are so if a hacker knows your behavior patterns then they might be able to break your brain wallet



And here is how brain wallets work you memorize these words and somehow they hash into this private key and the good thing about brain wallets is that even if you forget the private key and you remember the words you could just try many combinations of the words until you get the private key. the trade-off between hot wallets and cold wallets are security for the cold wallets but hot wallets are definitely more convenient because you don't have to memorize anything and they're just a lot easier to use.



Acquiring Bitcoin

You can get bitcoins from ATMs (some unique ones) just like normal money.

let's talk about exchanges there's two main types of exchanges there are centralized and decentralized, centralized exchanges are like trading different types of money on the airport you just trade between different types of currencies centralized exchanges are easy to use and they're supposed to be secure decentralized exchanges are usually more secure because they're anonymous and also it's harder for you to hack it because there's no central database that the exchange relies on. trades are person to person and some examples are bisque unit swap a big one is sushi swap and they're also trustless because you'll have to trust the other person to trade money usually how it works is that each exchange would mint a token which represents a certain amount of some sort of currency and you trade the currency for the token and then it goes into a single pool and then you can trade token to get the currencies that you want. so decentralized exchanges are more secure but centralized exchanges are more convenient because centralized exchanges tend to be easier to use while in decentralized exchanges you're anonymous and there's no single point of failure.



Create a wallet

By going to the site below you can now create a wallet.

<https://walletgenerator.net/>



the private and public keys



a paper wallet



a brain wallet

and to make a hot wallet we can go to the site:

<https://login.blockchain.com/#/signup>

Wallet mechanics

one feature of wallets is called multisig which just delegates multi-persons to have a certain key so in the regular bitcoin address each account has one key within the multi-signature address there are multiple signatures needed which is kind of l

What is a shared Bitcoin wallet?

Multisig is a cryptographic technique used in what we call a *shared wallet*.

A fact that is often confused or overlooked is that in a Bitcoin wallet, your funds are not inside the wallet - just as your debit card doesn't actually contain your cash. Like your debit card, you gain access to your funds through a kind of password (a very long, 78 digit password) called a private key. Private keys are held in your Bitcoin wallet, and without private keys, the associated Bitcoin cannot be used.

Basic Bitcoin wallets use one private key to access and send transactions while shared Bitcoin wallets require one or more private keys to access the funds connected with the wallet.

Private keys in shared wallets are often given to different people, called participants. For example, if you have three private keys, you might keep one yourself and give the others to family members. Using a shared wallet with multiple participants might seem unnecessary, but there are many benefits.

Why should I use a shared wallet?

The first key reason to use a shared wallet is that it is a solution to the Bitcoin wallet problem of having a single point of failure, which can result in losing access to your crypto assets. For example, imagine there is a fire in the apartment building you live in. Your computer and the paper backup keys for your Bitcoin are destroyed. Without the keys, you have no way to access those assets. But if your wallet is shared with others (who don't live in your building!), you'll still have access to your funds.

The other key reason to use a shared wallet relates to the utility that comes with having multiple decision makers. For example, you can introduce savings for your child by providing her with some funds in a Bitcoin wallet. If it's a shared wallet, you'll have the chance to review any transactions initiated before approving or declining.

How does a shared Bitcoin wallet work?

Recall that Bitcoin wallets do not actually contain bitcoin.

Wallets contain private keys that grant access to the bitcoin. In a basic wallet, there is only one private key connected to the wallet and that key is necessary to use the bitcoin. The private key is used as a mathematical signature to prove your ownership of the bitcoin.

In a shared wallet, multiple private keys are connected to the wallet. You will have to decide how many keys will be connected to the wallet, and how many keys will be needed to approve a transaction.

For example, if you decide to make a shared wallet with your mom and dad, there will be a total of three participants for the shared wallet. You decide that 2 of the 3 participants must sign a transaction for it to be approved (and thereby 'valid' to be broadcast to the blockchain). This shared wallet is called a "2-of-3 wallet." The shared wallet will have three private keys, but only two of the keys (in any combination) are required to approve transactions.

You can set the number of participants (up to 6) and number of approvals, such as, 1-of-2, 3-of-4, 6-of-6 etc...

How would this work in practice? Let's look more closely at the above '2-of-3' example. We are going to talk below about

setting up the shared wallet, so let's assume you and your parents have successfully created the shared wallet. Any private key holder can initiate a transaction through a **transaction request**. In this case, you, your mom, or your dad can request to move funds. As this is a 2-of-3 wallet, the transaction request will require just one approval from other participants, since the participant who made the transaction request (you) implicitly approves the transaction. If this was a 4-of-6 wallet, a transaction request would require 3 approvals. Imagine you decide to buy a new car with some of the bitcoin from your shared wallet. You call your father on his day off, informing him of your plans and asking him to approve the forthcoming transaction request. Then, you submit the transaction request. Your father quickly approves the transaction, satisfying the conditions of the 2-of-3 shared wallet. The Bitcoin is transferred from the shared wallet to the car dealership.

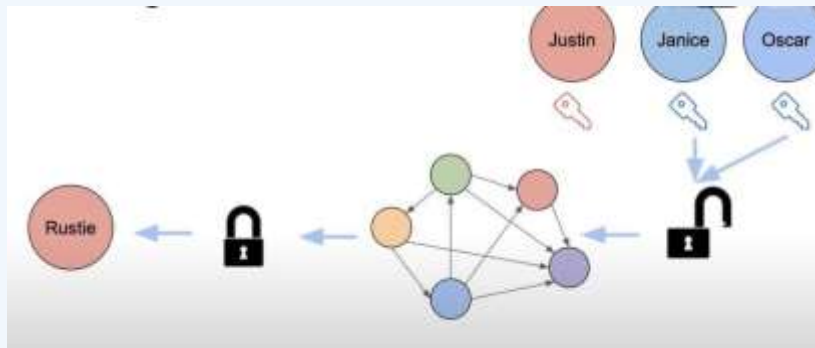
What are the downsides of shared wallets?

There are a few potential downsides to using a shared wallet. The first can easily be avoided by properly setting up the wallet. While it may seem that a 6-of-6 shared wallet is the most secure, this configuration in fact introduces an additional risk beyond a normal (single signature) wallet. Since a 6-of-6 shared wallet requires that all six participants approve any transactions, if even a single participant loses their private key, any funds in the wallet will be inaccessible. So, with a 6-of-6 wallet, you're effectively making the single-point-of-failure problem worse!

The other downsides of shared wallets relates to their ease of use and can be summarized as follows:

- A shared wallet requires making sure the other participants can perform the computer-related skills required.
- Once a shared wallet is set up it cannot be modified. Any modifications to a shared wallet, such as replacing a participant, requires setting up a brand new shared wallet from scratch.
- Transferring funds from a shared wallet will take more time than a basic wallet because, as described above, you have to wait for other participants to approve.

If you're looking to set up a shared Bitcoin wallet, you can do so in seconds using the **Bitcoin.com Wallet**. You can find more details on how shared wallets work and how to set them up in the



Bitcoin.com Wallet **here**.

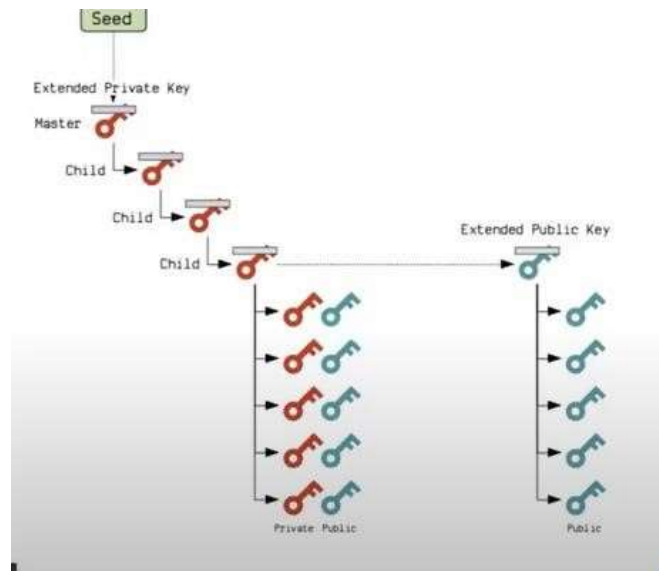
is let's say that rusty has a wallet but then he gives three keys to Justin, Janis and Oscar and if you have two keys then you could open the wallet so each of them they can't open the wallet but if two of them went together then they could open

the wallet let's say you needed four keys so that none of them could open the wallet but if rusty had four so there were seven keys in total and they each had three so in that case even if they grouped together they wouldn't be able to open the wallet but rusty would be able to open it if for some reason rusty lost one of his keys then he could ask either Justin, Janice or Oscar to loan their key so he could open his wallet you can see this in use by companies like coinbase and stuff where they might take one or two keys because they won't be able to access your wallet but then if you lost one key you could call coinbase and they could help you recover your wallet. **Hierarchical**

Deterministic (HD) wallets:

A hierarchical-deterministic (HD) wallet generates a new key pair from a master key pair for each crypto transaction to enhance privacy and security. Its hierarchical structure resembles that of a tree, with the master key “determining” the key pairs that follow it in the hierarchy. HD wallets have multicurrency support and can be restored with a recovery phrase. The vast majority of wallets are hierarchically deterministic.

they're deterministic because all keys are generated from a seed in the same way every single time, usually through a mathematical hash function which should give you the same answer every time you run it it's hierarchical because you could organize the key in this tree-like structure so what happens is you have your seed you hatch it once for your master and then you keep hashing it to have your child keys and then you also have your extended public key which you could hash again to get, which makes it easy to determine your keys you just hatch it through different hash functions each time that's good because there are fewer points of failure.



you start with a master private key and you can generate child private keys by hashing the private key plus the index of each of the wallets and you use the master public key of the master private key to generate all the child public keys and by the nature of hash functions the same inputs will always give the same outputs so you only need one key to generate everything because the outcomes are very predictable.

Mining:

A full-fledged bitcoin miner must:

- 1.Download the entire bitcoin blockchain.
- 2.Verify incoming transactions.
- 3.Create a block.
- 4.Find a valid nonce.
- 5.Broadcast your block.
- 6.Profit

Downloading the history of blockchain: you need to get blocks from your peers in order to see what blocks have been added in the past starting from the genesis block which is the very first block on the bitcoin blockchain and then you're

gonna stay up to date if you ever get broadcast someone's like hey look i found this block it's in your best interest to check it and then add that to your blockchain to make sure that you're always mining on the longest chain and not you know missing out on something that's happening by not adding and updating constantly.

Verify incoming transactions:

- listen to the Bitcoin network for transactions
- Unconfirmed(pending) transactions sit in the mempool for a miner to include it in a block.

-Verify incoming transactions.

the second step is to verify transactions and you're going to be using this mempool which is a term that refers to pending transactions that might be broadcast towards you and if you want to make a transaction on the bitcoin network you'll send it to some people and those people will send it to some people and those people send it to some people and miners in the network are going to be saving those unconfirmed transactions in their mempool and if they mine a new block they will grab from that mempool of transactions and verify that there's nothing bad happening with any of those transactions and make sure that those transactions check out and then they'll add those into their new block.

Create a block:

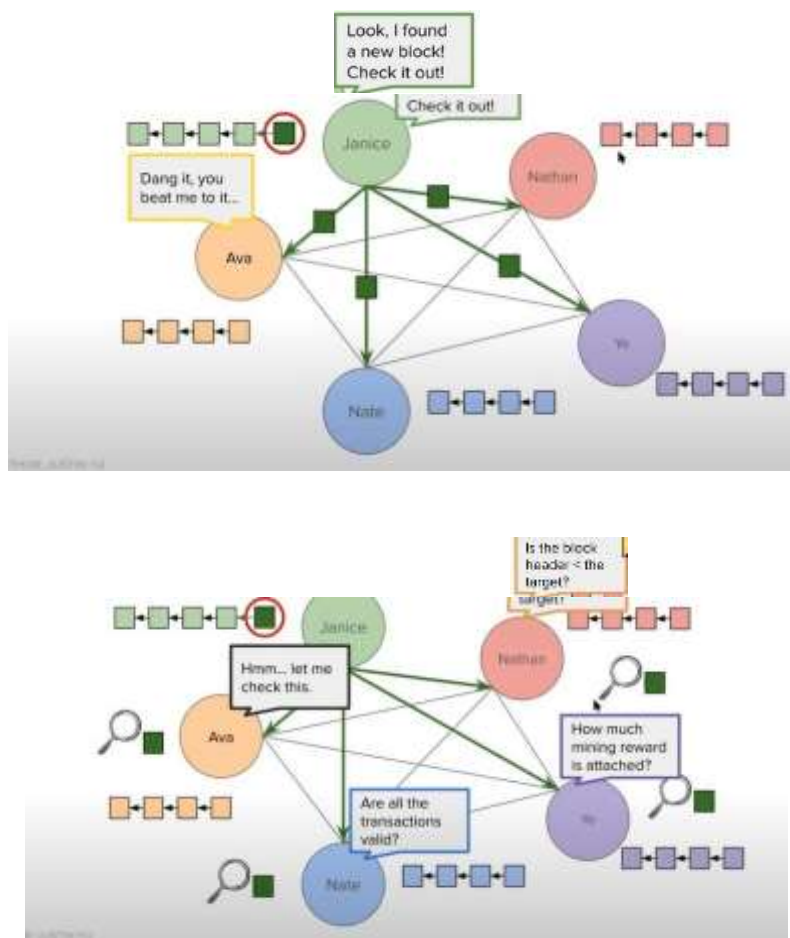
- gather transactions into a block (choose whichever transactions you want (most transaction fees))
- Get previous block hash and other necessary meta data(like merkle root and the block header etc.)

Find a valid nonce:

- find the proof of work.
- expand computational power

-Incrementing header nonce as necessary to change the puzzle.

Broadcast your block:



Let's say Janice was the lucky one to find that nonce she'll tell me and Nathan she'll say look at this new block I found she'll tell Nate and Ye and then we'll all check it out Nate's going to check if the transactions are valid Nathan's going to see if the block header is less than the target Ye is going to see how much mining reward is attached and we're all going to make sure that this was created with integrity and Jen didn't mess with the block at all and then imagine that I'm pretty pissed because I was trying to work on that block but it's not in my best interest to not accept Janice's block because the assumption is that the rest of the network will. if it's a proper block that was mined

correctly and then if I don't accept Jene's block I'm just gonna be left behind I'm gonna be mining on a shorter chain than everyone else. any blocks that I find let's say if I do find one on that chain won't be accepted by the rest of the network because it's going to be shorter than theirs.

Profit:

Know that mining is a competition for finding the next block.

The longest chain rules:

1-Block included in longest chain (you want to make sure that if you created a transaction that block is on the longest chain because the longest chain is how we determine which chain that we're gonna be working off of because at any given point in the network there will be different people with slightly different length chains so we always assume that that longest chain has the most computational power put into it and it has the greatest proof of work and that's the chain that we we're going to assume like is the right one).

2-Profit from block reward (coinbase transaction) and transaction fees (the block reward and the transaction fees are how miners receive profits).

3-All transactions added to canon transaction history.

Not included in longest chain:

1-Your block may not have been the first valid block seen by others.

2-start mining the next block (if your block is not on the longest chain that just means someone else found a block before you and they were able to broadcast it to more people in that amount of time and you just gotta move on to the next can't cry over it).

Mining incentives

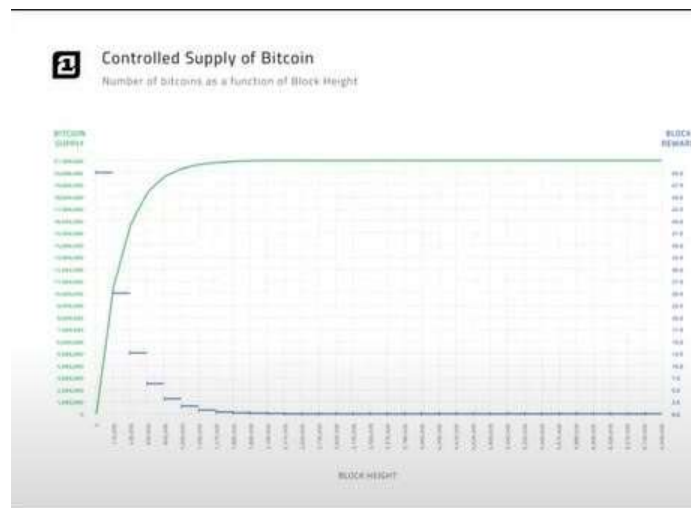
the main incentive for mining is profit.

Profit=revenue-cost. mining revenue is broken down into their block reward plus transaction fees and the cost of mining is broken down into fixed costs and variable costs (Mining revenue = Block Reward+ Tx Fees and Mining cost = Fixed costs + variable Costs).

if mining revenue is bigger than the mining cost miner will get the profit.

Block Reward:

Miner receives bitcoin for every confirmed block and the reward is 6.25 per block. when the miner is putting together the block and putting together the merkle tree the first transaction on the merkle tree is called the coinbase transaction and that's a special transaction to themselves which is incentivizing honest behavior like you're incentivized to make this block an honest block so that other miners on the network will accept it because if you don't then like you're losing out on your 6.25 in bitcoin if you want to ensure the fact that you make out with 6.25 at the end of this train then you want to make sure it's honest that other people will accept it as well and so that's another way that the network incentives align with the individual incentives. this value of 6.25 has every 210000 blocks and from every 210 000 blocks the value of bitcoin that is mine per block cuts in half and the next time we hit 210 000 blocks it's going to be 3.125 and so on and so forth. And also know that the bitcoin supply capacity is 21000000.



So to sum it up we learned:

1-Profit is primary motivator.

2-Higher incentive for honesty results in to more secure network.

3-Pseudonymous users results in to that there is no way to effectively track (or punish) dishonest behavior.

Conclusion:

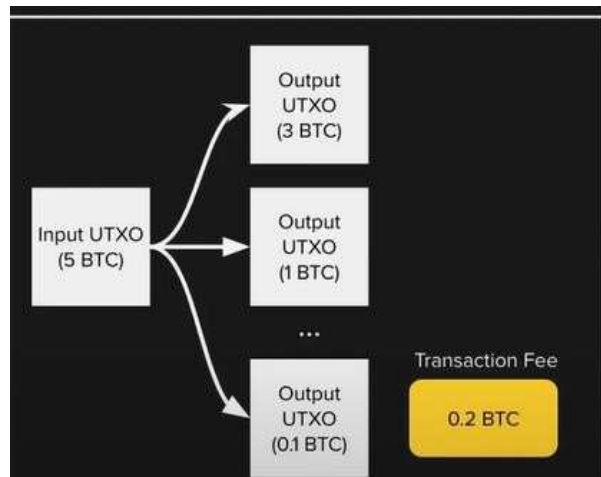
- Reward the honest nodes.

- Proof of work ensures that miners are dedicated to the network. **TX_FEES:** how are miners incentivized to be honest in their mining we've said that the incentives are aligned but why isn't it in their best interest to just mine an empty block to include that first coinbase transaction coinbase is the name of the exchange but they got their name coinbase from this transaction which is the first transaction in a given block where the miner sends 6.25 bitcoin to themselves and so why isn't the miner incentivized to just do this just put in 6.25 bitcoin and then broadcast that block to the rest of the network without putting anyone else's transactions? the answer is that mining revenue is two parts it's block reward but it's also transaction fees and if they were to mine a completely empty block they'd be losing out on a lot of profit that they'd receive from

transaction fees and when you input let's say five bitcoin and I want to send three to Nate one to Nathan and one to Janus and whoever the difference between the input and the output and the sum of all the outputs is equal to the transaction fee and so that's going to go directly into the pocket of the miner and the transaction creator so whoever's sending this these utxos sets that fee and that fee like I said can be set just by the difference between the inputs and outputs

it's not required you can create a transaction without a transaction fee but it's pretty much necessary because you can assume that other users on the network will be including transaction fees and your transaction will just not be included any block because every single miner will include other transactions over yours because they have zero to gain from adding your specific transaction to the block and this change actually just acts as an extra income for miners on top of their block reward so you can kind of think about it like the higher the transaction fee the faster your transaction will be confirmed because the more likely any miner in the entire network will pick yours up first because your transaction fee is higher than everyone else's and so they're going to receive a pretty high profit from that and like I said transaction fee is equal to input minus output so you have to sum all those outputs subtract it from the input and that will be the transaction fee. and this is really important because this is a popular question when the block reward goes to zero why will the bitcoin network continue to function won't all miners be disincentivized to continue mining? the answer is no because transaction fees will become the only source of revenue for miners and the conjecture right now is that transaction fees are going to continue to increase until we hit that block award of

zero and they'll act as the only profit for miners and only revenue.



how do we estimate exactly how much transaction fees to add to each of our transactions and there are three estimation algorithms all of the internet um and they basically take this history of all of the recent transactions they use their algorithm to figure out those transactions how many of them with this transaction fee were including the block how many of them with this transaction fee were included in the block and they can basically tell you the number of transactions in each block with these transaction fees so as you can tell obviously if you have a higher transaction fee you're going to be more likely to be added into a block and lower and lower and lower and so on and they group this pass transaction data into buckets and a bucket represents a range of transaction fees and these algorithms will spit out the lowest fee rate bucket where all transactions are confirmed it'll tell you historically if you use this transaction fee you'll never get stuck in the mempool you will always be included in a block and so that's what's gonna give you the lowest free rate bucket and that's a transaction fee that you should use to ensure your transaction is put into a block without wasting any extra bitcoin in the process but of course fee estimation algorithms aren't perfect and there's a

chance that you can use the algorithm use exactly the bucket that they tell you and your transaction still gets stuck in the mempool simply because everyone else that day decided to use a slightly higher transaction fee and yours just got stuck somewhere so sometimes we're going to need to bump up the transaction of an already broadcast transaction and something that I already sent now I need to say hey remember me like I will give you a little more money please include me in your next block, we only use this when the transaction gets stuck in the mempool this is kind of an edge case but we still want to go over it because it's still important and if the fee associated with contract the transaction is too low and no miner is going to pick it up this can be caused by a fee estimation algorithm that wasn't the most precise or a spike in fees there's a ton of things that can cause this and there's two different things that we can do to bump up our previous transactions the first one is called replace by fee(RBF) and child pays for parent(CPPP).

Replaced by Fee (RBF): in which the user who sent the transaction signs replacement transaction with the same inputs the same UTXOs that they sent into the transaction but now they're going to pay more transaction fees on top of that and it's going to come from a change in output because transaction fees equals inputs minus output so if you're using the same inputs then you have to decrease the output and so this transaction will have a higher likelihood of getting picked up because you're increasing the transaction fee.

Child Pays for Parent (CPP):

it's an interesting name for this but it's basically just referring to the fact that you're using this same one or more of the same outputs from the previous transaction that's stuck and you're going to create a new transaction with those outputs and you're going to attach a large fee to this transaction and the

reason this works is because miners package ancestor transactions with new transactions and so that's kind of like the child is gonna pay for the parent because this new transaction is going to grab that parent out of the mempool and say hey you're coming with me because we're going to be packaged together and my transaction fee is higher than yours an important note though is that for this second solution you need to have had at least one output pointed back towards yourself if I find bitcoin and I own Nate for bitcoin for the three teslas he bought me last week I'm gonna send him four bitcoin and then I don't need to send him five but I only have a UTXO with five in it right now so I'll send one back to myself and that's an example of an output that's getting sent back to myself and so in that transaction let's say that transaction gets stuck in the mempool i can say actually from that output that I just sent to myself I actually cut it down to 0.8 bitcoin and that'll be an additional 0.2 in transaction fees that I'll be giving to the miner and they'll be incentivized to include that transaction (that child transaction) and grab the parent one on the way.

Fixed costs: there's a hardware cost because all this mining is being done by using computational power and they need to purchase the hardware in order to execute all the all the functions of mining bitcoin.

	hashes / second	time to block (years)
CPU	20 million	7,620,101
GPU	200 million	762,010
FPGA	1 billion	152,357
ASIC	14 trillion	10.88

so the first one in the chart above is CPU, it is also the pioneer hardware this is what Satoshi directly references in the bitcoin white paper one "CPU one vote" that was the vision for bitcoin as we can see the hardware has developed a lot further than just CPUs but this would take 7620101 years to mine single block just probability wise if you were to use a CPU to try to find the nonce on the bitcoin network and then people turn to more intense hardware to try to mine bitcoin

GPU is an order of magnitude, faster than a CPU it's 200 million hashes per second which is references how many times the GPU can run that sha-256 algorithm on a random number and try to get that target nonce it can do this 200 million times per second this is a larger consumption of energy and higher production of heat than a CPU and this is super common in 2012 before FPGA and ASICs but there were a lot of disadvantages GPUs weren't specific enough to mining and they weren't really meant to be run in farms as mining farms became more of a thing side by side each other and since it's an ordered magnitude faster than a CPU it's going to take 762 thousand years to mine a block which sounds like a long time but it's still faster than CPU.

FPGA which stands for *peeled pro field programmable gate arrays* and that's when we start to see actually bitcoin specific hardware being developed without losing all customizability this is an exclusively bitcoin but when hardware more tailored towards mining started to be developed and you get a little bit of a trade-off between just dedicated SHA-256 algorithms which is what we see in ASIC which is exclusively meant to be to just run SHA-256 which is why it's so fast but it also can literally do nothing else and general purpose hardware.

So there's this trade-off if bitcoin fails then SHA-256 specific hardware will become worthless like an ASIC but if bitcoin

thrives then that specialized hardware is going to get you further and it's going to generate a higher profit.

ACIS is the final hardware that we've seen in the in the mining space this stands for *Application Specific Integrated Circuit* this does nothing but SHA-256 it's going to 14 trillion times per second which when you compare that to 20 million is just a really high number and it makes it better than anything else, there's a huge variety of ASICs with various trade-offs in general though there's a lower base cost and lower electricity usage and it's pretty small but it's decently compact with a higher hash rate but it can get pretty expensive for example an amp miner s9 is 1500 so this is definitely like what we refer to when we say you have to be dedicated to the network this is your fixed cost if you actually want to get serious about mining bitcoin.

Variable costs:

variable costs which is the second part and we'll think about the different types of energy that we consume when we're mining bitcoin:

- the embodied energy which is kind of wrapped up inside purchasing that hardware because it was used to produce your hardware.
- electricity which is what we use to power your hardware. -
- cooling to maintain your hardware we need to cool down that hardware because if you use it too much like it might just break down so you got to cool that hardware to maintain it you might even need infrastructure like warehouses if you're running a mining farm you need personnel to execute like all this hardware for you and all that energy gets converted to heat and there's a lot of energy that's being consumed here and this is kind of why we need these fans there.

Lecture 5

Bitcoin in the wild: game theory and attacks

<https://www.youtube.com/watch?v=6qfoylDftK8&list=PLLkiRvMHnp6OIWkZVa85g2tv7NtlgoAID&index=5>

Mining pools

Probably shouldn't surprise you to hear that a mining pool is a way for individual miners to combine, 'pool' their computational power together and what this effectively does is reduce the variance in your expected reward for mining (when you're solo mining more or less what your probability of finding a given block is and you can sort of get like your expected value from that but there's a lot of risk that you're taking on, what you do is you pool together, you band together, with a bunch of other miners and you communally reduce the risk you get, more consistent payouts and there has to be some entity operating this sort of coordinating the different miners and assigning payouts). there is a pool manager or pool operator, this tentative will take some cut of the mining rewards. you

might ask yourself how these rewards get paid out? the mechanism by which we do that is keeping track of something called shares, when you're mining for an actual valid block you're done when you find a hash with some sufficient number of preceding zeros, this is given by the difficulty parameter and means that I had to get random numbers enough times such that one of them that had this many zeros and that's how we know you really spent that much work because you had to crunch out a lot of random numbers to get one that started with that many zeros, the same idea still holds in mining shares but now we're getting less of the leading zeros so it's essentially like solving the same proof-of-work algorithm but with a lower difficulty. this way we can still prove that the miners in the pool are doing work but we're not just tracking every single successful block that they produce because individual miners are not likely to produce an individual block this is how we keep track of how much work you're doing for the pool. you'll get paid out for every valid share that you submit. there's a couple of different ways that these payouts actually happen, a couple different schemes for this, that we're about to go into but now we're sort of forfeiting the idea that if you're the one who finds the block you get the coinbase reward the currently 6.25 bitcoin minted in your name. even if you find a valid block is the same thing as a share, it still satisfies the conditions for validity for a share and it's treated the same so you don't get that money. that's like you just get paid the same as any other share that you would have submitted and that reward is distributed across the pool. how might these rewards get paid out?

There's tons of different derivatives schemes of how we can pay out miners and try to align incentives really well but the two basic ones uh that were initially thought up were called the

proportional reward schemes and a pay-per-share scheme which we'll start with the pay-per-share scheme.

every time you submit a valid share you just get paid out some constant amount and this constant amount is dependent on the current difficulty of the bitcoin network as a whole and the relative difficulty that pool is requiring for shares. but every single time you submit a valid share, no questions asked, you get some flat rate, everyone else gets the same flat rate for submitting a share, and this is great for the miners, they have no risk anymore, they're no longer like won't I find a block and get a reward. no you submit a share, you get a reward said and done.

there's the risk of when will you find a valid share but the difficulty that pool set is sufficiently low that this really lowers the barrier for consumer level hardware or consumer level amounts of money to spend on specialized hardware.

the pool is taking on all the risk, regardless of whether or not the pool collectively finds a block it's still paying out its miners at every share so the pool has to keep a reserve of previous rewards that it can pay out from and it also stomachs the risk of like I'm paying out from these reserves until we collectively find a new block but the issue here is that you're not incentivized to submit a valid block, you could try and get fancy with how you're mining for this pool and submit valid shares but when you find an actual block you try to withhold it and submit it to the blockchain on your own because you're still getting paid for every single share so why should you submit a block.

the other approach is something called a proportional reward scheme which is when the pool will only do these payouts, not on every single valid share that you submit, but rather when we collectively as a pool find a new block.

we collectively as a pool find a new block, remember all the miners in this pool are solving the proof of work algorithm for the same block at the same time, they're all still racing to find the same block but now they're just splitting the rewards. now in this scheme when we collectively find block the payout actually happens and how much you get paid is proportional to how many shares you submitted in the search for this block. let's say okay we just found a block and we're starting the search as a pool for a new one together and let's say collectively across, everyone that's in the pool it took us 100 shares, and then we found the block and maybe I created 20 of those shares that means I'll get a fifth of the reward for that block. this is more beneficial for the pool because now we're not paying out of some reserves, we're only paying out when we find a block so we're not really bearing any risk. of like we have to pay out of pocket to these miners in advance, individual miners now will bear some of this risk of like I'm sort of waiting around till all of us find a block and that's a little less risk than if I was solo mining but still not born entirely by the pool anymore but the larger the pool is the more consistently you will find blocks. this is a lower risk for pool operators for the same reason that it's a lower risk for the pool.

Pool pays out **when blocks are found**. Miners are paid proportional to the number of shares they've submitted since the last block.

- More beneficial for the **pool**
- Individual miners still bear some risk in variance proportional to size of the pool
 - Not a problem if pool is sufficiently large
- Lower risk for pool operators - only pay out when reward is found

let's take a look at numerical terms why one might want to pool mine instead of solo mining? now so sorry that should say um the date like this wednesday's date it's not from uh

11:30

last year this is a slightly old slide but the network hash rate in (10/06/20) is at about 155 half million tera hashes per second which is a cosmic amount of hash power but if you were to buy a sort of single mining rate, top of the line mining rig, the Antminer S17 that's getting you 56 tera hashes a second, it's somewhere like 3k USD to buy. the current blocker reward is a 6.25 bitcoin right now and so what that means is that your proportion of the sort of total network hash rate, the of all the hashes produced in a second, you produce. The rate will be 3.6 times 10 to the negative seven that's how much of the network's hash power you control which is not much. this is more or less like your probability of finding the next block because you imagine like every hash is a perfectly random throw at the dartboard you're putting up 56 of the 155.5 million dart throws a second so that's like about your probability of finding the block.

- Today's (10/06/20) network hashrate: **~154,582,000 TH/s**
- Antminer S17: **56 TH/s**
- Current block reward: **6.25 BTC**
- Proportion of network hashrate = $(56 \text{ TH/s}) / (154,582,000 \text{ TH/s}) \Rightarrow \mathbf{0.00000036}$
 - ~ your probability of finding the next block

Now let's take a look at our expected reward for solo mining.

Solo mining:

- $EV = 0.00000036 * 52560 (\sim \text{\#blocks/yr}) * 6.25 = 0.118 \text{ BTC/yr}$
 - $EV[6.25 * \text{Binomial}(N=52560, p=0.00000036)]$
 - However, $SD = 6.25 * \sqrt{(N * p * (1-p))} = 0.86 \text{ BTC/yr} \Rightarrow \text{high variance (risk)}$
- Expected to mine -1 in every 2,777,778 blocks $\Rightarrow 52.8 \text{ yrs}$
 - $EV[\text{Geometric}(p=0.00000036)]$
 - Not to mention halvings & changes in hashrate
 - Last semester, this number was 46.4 yrs...

6.25 that's the bitcoin reward and let's multiply that times the expected number of blocks that I find in a year and there's assuming that the difficulty like metric is well maintained and the block is produced about every 10 minutes, there should be 52 560 blocks a year. we're just looking at the expected value of successes in 52 560 trials where the probability of a success is that we computed before, your proportion of the hash rate, if we take that and we multiply it by the expected reward per block that means we're looking at 0.118 bitcoin a year. that's just our expected value but if we take a look at the standard deviation of this it's 0.86 bitcoin a year.

maybe these numbers don't mean much to you but when we compare them we'll see that this is actually really high risk and you can see it's high risk. for example if we're saying we expect to make 0.118 bitcoin a year but the standard deviation is almost eight times that value, there's a lot of variance that we're taking on, not to mention that expected value is a bit of a brute metric, if we were to take a look at the expected time to wait until the first block that we find. taking this and making it a geometric random variable with that probability, this is the expected time that we have to wait until the first block is found and it comes out to about 52.8 years. if we start to factor in things like the expected price trajectory of bitcoin and the number of paddings and changes in hash rate then we're going to assume hash rate increases which means that your

probability gets worse and you are not going to find your first block for a while if you're solo mining. last semester that number wasn't 52.8 years it was 46.4 so the hash rate is growing quickly.

let's take that same Antminer S17 and let's join a pool with it what's our expected reward?

Pool mining:

- Assume pool has $\frac{1}{6}$ network hashrate
- Your proportion of pool hashrate = $56 / 25,763,666 = 0.00000216$
- $EV = 0.00000216 * 52560 * \frac{1}{6} * 6.25 = 0.118 \text{ BTC/yr}$
 - $EV[0.00000216 * 6.25 * \text{Binomial}(N=52560, p=\frac{1}{6})]$
 - **Same EV as solo mining! But:**
 - $SD = 0.00000216 * 6.25 * \sqrt{(N * p * (1-p))} = 0.0012 \text{ BTC/yr}$
- Expected to mine -1 in every 6 blocks => 1hr
 - $EV[\text{Geometric}(p=\frac{1}{6})]$
 - Much more consistent revenue stream

let's assume that the pool has a sixth of the network's hash rate, this is not a wild assumption there are a couple of pools with around this the size of hash rate, now we're taking a look at the pool, we're assuming this pool is in the proportional payout scheme, so your pool would pay you proportionate to how much hash power you contribute to the pool, if we take your proportion of the pool's hash power that'll be your proportion of the pool's rewards and that's 2.16 times 10 to the negative six, in order of magnitude better than the previous p value that we were working with, now when we take our expected value, we're now taking our proportion of the pool's compute times the expected reward of the pool as a whole for the year and you notice that our expected value is actually exactly the same as when we were solo mining but if we were to take a look at the standard deviation of this, it is two orders of magnitude smaller than in the solo mining reward so we're taking a look at the same expected value for two orders of magnitude and less risk. pool mining is really the move for individual miners and here if we're to take a look at the

expected time to the first success, expected time till the first block mined by the pool, when you've got a sixth of the pool's power you're expected to do one in every six blocks so if you're mining a block every hour that's a pretty consistent revenue stream even at a high level the point here is that it's really worth it for individual miners to join pools, it's much less risky for them to do this.

the pro and con

pros are that pools allow individual miners to participate which is a big step in the direction of one CPU, one vote this should be truly decentralized to anyone with a computer the cons for that we introduced this sort of trusted entity of the pool manager and there are ways to try to mitigate risk in trusting the pool manager and there's a number of attacks and there's a number of ways to game mining pools in a way that's not necessarily beneficial to the bitcoin network.

+ Pros

01 Allows individual miners to participate

- Cons

01 Pool manager must be trusted

02 Enables a multitude of attacks

is it possible that mining pools aren't perfectly incentive aligned? can we maybe take advantage of some incentive misalignment?

The answer is yes and we could say that they're susceptible to some amount of gaming.

Now we'll introduce a concept called pool hopping. let's take a look at those two reward schemes that we painted, the first is the proportional rewards where every share that you submit are inversely related to the number of shares submitted while mining the current block. what we're saying is this, let's say you're in a proportional pool and you just started the pool, just started mining on a new block, and let's say suddenly you submitted one share and it was a valid block that means it took the pool one share to find that block you submitted. that like one of 100 shares submitted for that block and so you get hundredth of the rewards. Similarly if the pool is taking a while to find this block and let's say we're at 50 shares you've only submitted two and the pool is still crunching out shares, the more the shares submitted for this block, the greater the denominator grows on your claim to the rewards for that block which is why we see this red curve forming in the graph below.



let's say I'm in a pool and we're going from zero to 25, 50, 75 of shares that we as a pool are submitting before finding this

current block, the red curve lays out what the resultant rewards per share I will get.

um no matter how many shares I put up for the pool as we get onto 50 shares, saying at this point I get 0.25 bitcoin per share, when we contrast this to the reward per share or the pay-per-share pool the rewards per share are constant because the whole idea is no matter what every time you submit a valid share we just give you a flat rate, so the question is if I'm an individual miner and I'm just trying to maximize profit, how would I do that? the idea here is to spend your time where it is the most valuable, let's say block just got put on the bitcoin blockchain so the idea is you want to start off by mining in this proportional pool and you're starting up on this red curve because let's imagine that in 25 shares submitted to the pool we do find the block and I was still in that pool when we found the block having submitted some number of shares for that small amount of total shares, I'll get this high reward per share, so I want to stay in the pool but the longer it takes our pool to find a block, the smaller piece of the pay that I get so I'm going to spend my time in this pool until the point where we collectively have produced so many shares that I only account for a small proportion of that and my reward per share actually becomes equivalent to what I would get if I was just in a pay-per-share pool. it's actually optional for me at that point to step out of the proportional pool or hop over to a pay-per-share pool because at this point I'm getting as much value out of my time, better value out of my time in a pay-per-share pool because I'm no longer dealing with the pool as a whole keeps submitting shares and I'm getting a smaller and smaller piece of the pay.

pool cannibalization

it's a strategy in which rather than taking your computational power as a pool and using it honestly (solving proof of work, trying to get these blocks) what you do is you say okay my miners go crunch out proof of work at some other pools, make shares for them but withhold valid blocks never ever contribute to their revenue (it's like should you ever find a valid block you bring that back to me). it's nice because your miners will still receive rewards from those pools and it's also undetectable by other pools unless it's statistically significant that they're being cannibalized like to them it looks like some miners joined and they're making shares but I can't really tell that, they're withholding blocks from me.

it's more worth to do this cannibalization tactic than to pool mine honestly.

Pool cannibalization breakdown let's say that we're a pool and we have 30 hashes per second and if we take a look at the whole network (we'll think of the network as comprised of multiple pools), all of it together is 100 hashes per second and we're gonna say that the current block reward is one bitcoin for simplicity. using the same style of expected value calculation as before we've got 30 of the hash rate that's pretty much our odds of finding a block, that means our expected value is 0.3 times the block reward (we're looking at 0.3 bitcoin per block) that's our expected value so that's sort of our baseline as a pool.

Standard mining strategy:

Add 1 H/s to your own pool

Your hashrate breakdown

- $31/101$ of network H/s = 30.69%
- Reward: $0.3069 * 1 \text{ BTC} = 0.3069 \text{ BTC/block}$
- Marginal revenue = **+0.0069 BTC/block for +1 H/s**

now let's say we've got another miner looking to join and they've got one hash per second of computational power that they're bringing with them. first let's take a look at the standard mining strategy which is to say hey come on in and mine for the pool and the result of that is that now you've got 31 hashes per second out of a total of 101 in the network, because we're making the assumption that this miner did not come over from a different pool, this miner was not already accounted for but rather they joined the network sort of fresh. we take proportion of the hash rate, multiply it times the expected or times the reward per block and we get an expected reward of 0.3069 bitcoin per block so you can see we've got our marginal revenue point nine times ten to the negative two bitcoin per block for an additional one hash per second to the network and that's sort of like the marginal revenue of one hash per second that is point zero zero six nine.

Pool cannibalizing strategy:

Distribute 1 H/s among all other pools, **withhold valid blocks** (submit back to your pool)

Helpful to think of the rest of the network as a single pool

- Pool thinks it's running at 71 H/s, you're getting $1/71$ of the reward
- However, only 70 "effective" H/s, as your 1 H/s will never result in a valid block being submitted to the pool
- $EV = 1/71 * 70/101 * 1 = \mathbf{+0.0098 \text{ BTC/block for +1 H/s}}$
- **42% larger** (expected) revenue gain from pool cannibalization

but can we do better with the pool cannibalization strategy so here again what we do is that we say to this miner okay you're one hash per second, send that one hash per second (the share that results from that) to a bunch of different pools but should any of these hashes ever result in a valid block again you withhold that and you bring that back to our pool and here it's actually helpful to think of all the other pools in the network as one single other pool and this would mean that this single other pool has like over 51 percent of the hash power which is why I want to again reiterate that these are a series of other pools we can look at them as equivalent to one. if we look at this single pool (the rest of the network) it's getting 71 hashes per second because it's got your miner submitting that extra one hash per second of shares to it and because it thinks you're doing so honestly, you're clued into one over 71 of the reward (that's your proportion of the hash power for that pool) however this pool is only getting an effective we'll call it 70 hashes per second and the reason why is because that this one hash per second that you're contributing to this pool is never going to result in a valid block being submitted to the pool. that work is wasted because it never results in a block being created from the perspective of that pool that you're cannibalizing so your expected value here is now you own 1 over 71 of that pool so you get 1 over 71 of that pool's reward but again there's only an effective 70 hashes and that this pool is putting out to the network so it's putting out 70 over 101 of the total hashes in the network and you're getting one over 71 of that times the one bitcoin per block so now you're looking at point zero zero nine eight bitcoin for block for an additional one hash per second and this is forty two percent larger marginal revenue than the standard strategy.

Pool cannibalizing strategy:

Distribute 1 H/s among all other pools, **withhold valid blocks** (submit back to your pool)

Helpful to think of the rest of the network as a single pool

- Pool thinks it's running at 71 H/s, you're getting 1/71 of the reward
- However, only 70 "effective" H/s, as your 1 H/s will never result in a valid block being submitted to the pool
- $EV = 1/71 * 70/101 * 1 = +0.0098 \text{ BTC/block for } +1 \text{ H/s}$
- **42% larger** (expected) revenue gain from pool cannibalization

Pool wars we can see this is incentivized and we can even model this as a simple game where there's two players, pool one and pool two and in this game it's a simple iteration where at every turn each of the players get to make a simple decision, do I mine honestly or do I pool cannibalize. this boils down to the prisoner's dilemma (the idea is you and your boy just pulled off a heist or something but you've gotten caught and the cops split you up, you can't communicate with one another, you're both getting interrogated and the cops give both of you this deal and they let you know that the other person is getting the same deal they say you turn your boy in and you go free or they get life same thing happens if your boy turns you in but you stay silent you get life they go free but if you guys both turn each other in you're both going to be here for like 10 years or something like that we can expect that most both prisoners will end up choosing to rout one another out and it's because that choice ends up being something we call nash equilibrium.

- Attack decisions resemble an iterative game
 - Two players: Pool 1 and Pool 2
 - At each turn, the player chooses whether or not to attack the other
- Each iteration of the game is a case of the Prisoner's Dilemma

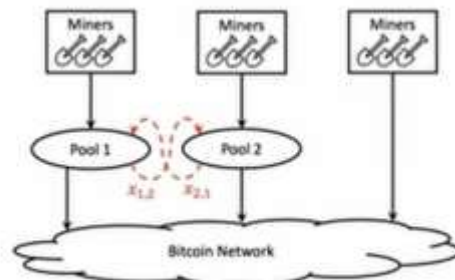


Fig. 7. Two pools attacking each other.

	POOL 2 IS IDLE	POOL 2 ATTACKS
POOL 1 IS IDLE	POOL 1: +3 POOL 2: +3	POOL 1: +1 POOL 2: +4
POOL 1 IS ATTACKS	POOL 1: +4 POOL 2: +1	POOL 1: +2 POOL 2: +2

Nash equilibrium is a game state in which no player can earn a higher reward by **changing** strategies, assuming all other players' strategies remain **unchanged**.

let's take a look at nash equilibrium from the perspective of these pool wars, so what you're looking at here is a payout matrix and this describes a set of outcomes in this game that is not unlike what has been described in the prisoners dilemma. one choice that we can make is we both mine honestly, we'll get some profits for that we'll call that like plus three units of profit. now let's say one of us attacks while the other mines honestly, that's either bottom left or upper right square and in that case the honest pool is getting some amount of its rewards siphoned from it and the other pool is getting that additional marginal revenue that we're talking about, so we're going to call this like a plus four for the attacking pool and a plus one because it's still getting some revenue for the idol pool and then finally we have the situation where both pools attack one another they're siphoning funds off from one another and trying to snoop this marginal revenue of each other, so we'll call that a plus two situation. if both pools stay IDLE and mine honestly and don't siphon funds off of one another but we can't

actually expect this to end up happening and here's why let's assume that the other pool whatever they're doing they'll maintain that strategy and let's say we're both IDLE and I'm assuming pool 2 maintains their strategy so I'm assuming pool 2 stays idle and then I ask myself the question would it be better for me if I changed my strategy? well let's see assuming pool 2 maintains and I change then I end up going from plus 3 to plus four and pool two goes from plus three to plus one we don't really even care about that we just see that it is better for us to change our strategy assuming pool two maintains. we've just decided assuming pool 2 stays IDLE and it's better for me to go on the offensive so now let's assume that's the situation we're in I'm on the offensive and at this point if we're taking a look at pool 2 now from pool 2's point of view it's assuming pool 1 maintains their strategy that's sort of the assumption that runs here (assuming the other player maintains) so pool one is on the attack now and they're going to stay on the attack is it better for me to change my strategy? if I don't change my strategy I stay being attacked and I stay earning plus one but if I change my strategy, I go from plus one to plus two and now finally we're in the state where we're both attacking one another and if we do the same assumption, assuming the other pool stays attacking me is it better for me to switch my strategy to honest mining? no because I'll go from like a plus two profit to a plus one and so this is a state in a game that we call nash equilibrium where basically for all players we have the condition that assuming that every other player maintains their strategy I can't do better by changing my own so it's better for me to stay where I am assuming everyone else stays where they are and we only find ourselves in this nash equilibrium state when we're both attacking one another same thing in the prisoner's dilemma is when they're both ratting each other out.

pool cannibalization ends up being incentivized. *Forking and double spends*

Bitcoin core bitcoin core is an open source project that works on full node software and it helps to fully validate the blockchain and also bitcoin wallets, what this means is because bitcoin itself is decentralized this is a group that works to make improvements to the system and gradually help by tweaking the protocol from time to time, the software designed by these developers is used by full bitcoin nodes.

- **Bitcoin Core:**
 - The team of developers in charge of the Bitcoin GitHub repo
 - The software designed by these developers used by full Bitcoin nodes



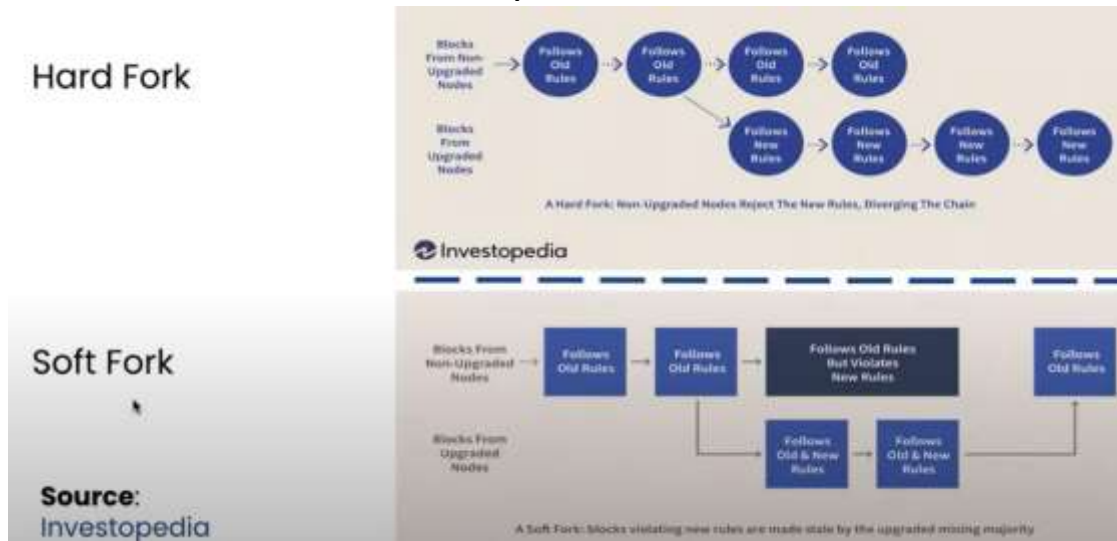
Bitcoin improvement proposal this can refer to changes to network protocol, block or transaction validation or other, anything that affects interoperability (ways that we can interact with bitcoin itself). a lot of times we have these three standard types of improvements that we can make which are standard track, informational and process forms of bitcoin improvement protocols and these are gradually improve the bitcoin network.

- **BIP: Bitcoin Improvement Proposal**
 - Three types:
 - Standards Track BIPs
 - Informational BIPs
 - Process BIPs
- First BIP proposed by Amir Taaki on 2011-08-19
- Signal support for a BIP by including reference in block when mining



Source:
https://en.bitcoin.it/wiki/Bitcoin_improvement_proposals

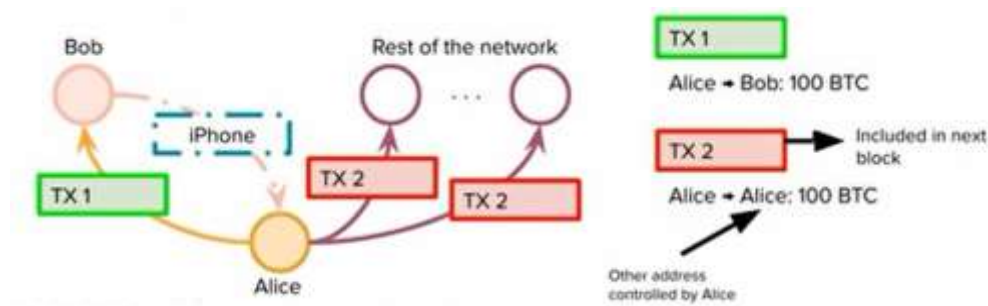
Forks there are two types of forks and forks are just times when we see a divergence from the blockchain network so normally when we think of a blockchain network, it's just a chain and we kind of envision it as being like a long chain of individual blocks but sometimes we do have these forks which is where we see like a split in the network.



soft forks are backwards compatible and it means that you can continue to use the same protocol after it has been bought and you don't necessarily need to update but let's say right now you're mining or working on the bottom chain on the picture above you can work your way back and continue to work on this old protocol and continue to do so for, as long as you want whereas in contrast we have hard forks where once we have a fork we can't really be traced back and work from the block that the fork has happened after it. one example of this is bitcoin cash, bitcoin cash was like a really early version of bitcoin itself, and because it was a software you can continue to work under bitcoin cash but it's not that commonly used anymore, if you really want to, with soft forks and bitcoin cash which is part of segwit you can continue to work on that same chain. **Double spending**

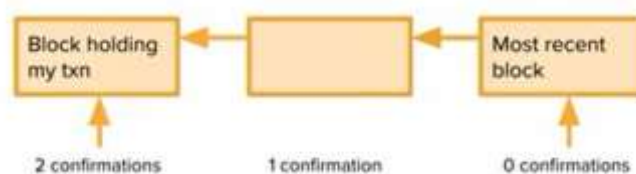
we can think of double spending as the of sending the same bitcoin or cryptocurrency more than once.

let's just propose a little situation here, the year is 2099 and Alice wants to buy an iphone 92 from Bob on the black market for 100 bitcoin but doesn't want to give up her bitcoin so how can Alice do this? acquire this iphone without having to spend her bitcoin. for starters they can do something called a race attack, a race attack is essentially where you try to move faster than network itself and you want to move at a speed that allows you to pull a fast one on the rest of the bitcoin network.



the race attack here is where we see that Alice proposes some transaction and tells Bob well I'm gonna give you 100 bitcoin you should just send me your iphone and while we have this transaction that is being sent to Bob from Alice, Alice says to Bob I'm giving you my 100 bitcoin at the same time Alice is telling the rest of the network that she's directing the bitcoin to someone else and so being very deceptive about where the bitcoin is being sent and the reason why this is called a race attack is because it's all done really quickly. let's say that Bob naively saw this transaction and assumed well I have my hundred bitcoin now it's safe to send the iphone, the minute Bob sends the iphone that timeline might not correspond with the actual sending and block broadcast verification process even though Alice told bob she's going to send a bitcoin.

It is a race attack because due to time it can be quite problematic for Bob because you don't know exactly when you are ready to receive your currency. **Confirmation** this is where we introduce the idea of something called block confirmations, block confirmations is just waiting for spending some time and waiting for more blocks to be added to the chain before you know that your block is a valid transaction and secure, this introduces a little idea of block depth. if you look at this block on the left in the picture below.

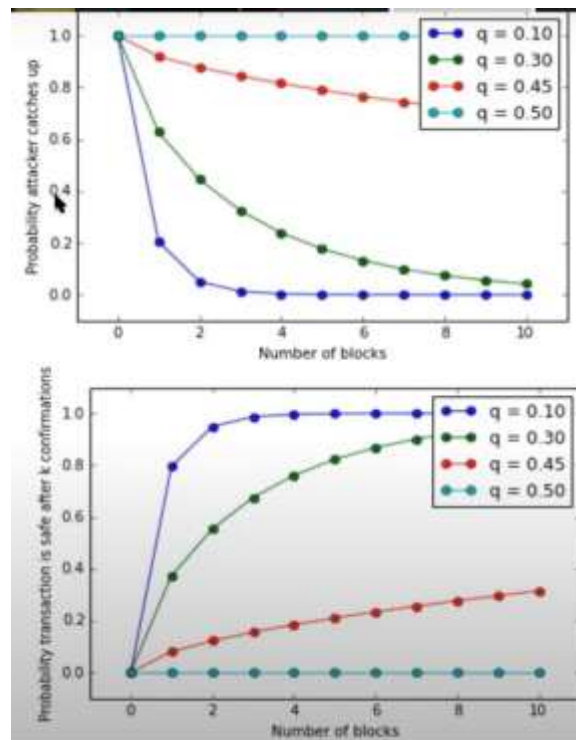


let's say that this contains my transactions so this contains the 100 bitcoin transaction from Alice to Bob so there are two confirmations on the right of it and you should know that the more confirmations you have the more sure you can be that the actual transaction is valid and has gone through and has been approved by the network, this is where we see the idea of block depth where the greater depth you have, the more confirmations you have built off of your original block, the more sure you can be that it's actually going to be approved and run properly.

Block Depth

A block's position index in the blockchain relative to the latest (most recently added) block. A block that is five blocks before the latest block will have a block depth of 5.

in order to understand this let's go into a little more about confirmations and why we can know that a transaction is confirmed, there's a lot of probability involved with block confirmations and transactions.



The upper graph shows probability that the attacker can catch up and try to perform a race attack and pull a fast one on you here and each variant color shows different computational power than the attacker might have, it shows the percent of the network that attacker has.

let's say in the case that an attacker has 10 of the network hash rate if we wait for six block confirmations, we can be sure that our transaction is reasonably safe so after six block confirmations we're pretty overall confidence that you can be sure that your transaction is valid. the bottom here which is the inverse shows the probability of a safety vendor given the assumptions of attacker hash powers, if we assume certain hash powers, we can assume that our transaction or overall circumstance will be safe so as we see with lower hash powers

it's generally safe to say that after six confirmations there's a very low likelihood that an attacker can double spend. but if an attacker has 50 of the hash power we can never really know for sure that the vendor is safe while we have the idea of block confirmations, a key component is hash power and if a certain attacker has too much of the network then it can just like a 51 attack the attacker can do whatever they please once they have the majority of the network.

51% Attack

let's say that Alice is malicious and Alice owns more than 50 of the total network hash power, so whenever Alice's chain is behind the honest network's chain she will always be able to catch up and out-produce the honest miners so she'll always be able to perform this meaning that if Alice has more than 50 hash power she can successfully double spend 100 of the time which is why any one party having more than 50 of the network pretty much means a full monopoly on of the everything going on in the network. a few questions that might come up with is that why would Alice not want to double spend and you might think well if Alice has more than 50 of the network hash power why would she not want to double spend and a lot of this comes down to overall confidence and value of your currency, if the rest of the network detects double spend then the network loses confidence. imagine that bitcoin we hear on the news that bitcoin somehow someone was able to get 51 of the hash power and controls all of bitcoin, what do you trust something that was initially meant to be decentralized but instead is now so much more monopolized?

there are a few different reasons if your overall confidence of cryptocurrency decreases and diminishes then you won't trust the currency and no one will really participate in it as much, let's say if the us government kind of went down the toilet and

didn't necessarily have any value related with a dollar or whatever currency then there's no way that people would want dollars. the whole point of a currency is it has value and widely accepted value and it's important to keep the value of any cryptocurrency high that everyone needs to have trust in it and believe in it.

another thing that Alice can do is to bribe miners and she might not have all of the physical hardware like the ASIC which is like the mining hardware involved to perform a double spend and to actually successfully do this but instead Alice can bribe miners or entire pools in certain circumstances. if you know that you will be rewarded greatly then you're pretty incentivized to help Alice out and work for Alice because you will get guaranteed payout once you get past 50.

Why would Alice not want to double spend?

If the rest of the network detects the double spend, it is assumed that confidence in the cryptocurrency and exchange rate would plummet.

Bribing Miners:

Alice might not physically control the mining hardware necessary to perform a double spend.

Instead, Alice can bribe miners or even entire pools to mine on her withheld chain.

Bitcoin: The New Gold Standard

imagine if Alice is some kind of large institution, a government altcoin, whatever it is but they have a lot of resources or capital available so that they can actually make this a reality so Alice can acquire enough ASICs and participate in bitcoin or in crypto in general to achieve this greater than 50 effective hash power. right now we see what is called a “goldfinger” attack, the idea in this bond movie is the gold standard is what a lot of fiat currencies use to measure value of their currency and if gold is more valuable by itself then you'll overall have a greater wealth

and you'll be much better off, if there's a way hypothetically in this movie they lower the amount of gold possible, they actually blow up or they turn a large sum of gold radioactive so that the gold that goldfinger owns is more valuable. it's called a goldfinger attack is because it destroys the target cryptocurrency by destroying the confidence with a double spend or spamming the network with empty blocks.

Censorship

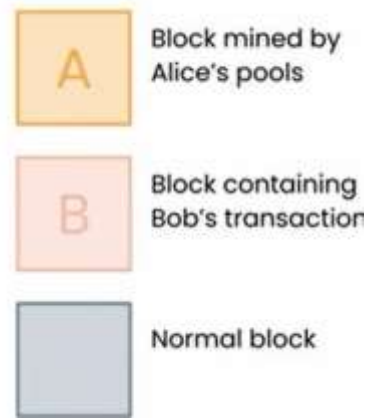
a lot of what we're talking about, the attacks, the manipulations are deeply dependent on what we call bitcoin's fork resolution policy.

how do we know which chain to trust? well it's the longest one when you have 51 of the computational power, as you look out to a very long time range you're basically guaranteed to be able to make the longest chain. we're going to try to make the longest chain or mine on top of it for as long as we can and it also ties into how one might try to censor someone else which means that make sure that their transactions never end up in the longest chain. let's set the stage.

we've got blocks mined by Alice's pools where Alice is a government or maybe someone that has control over a government or a jurisdiction or a lot of mining pools. the point is that Alice's mining pools control over 51 percent of the network's hash rate.

Say Alice is a government, or has control over a government, that has jurisdiction over mining pools. In addition, Alice's mining pools control **over 51% of the network's hashrate.**

Objective: Censor the Bitcoin address owned by Bob, and prevent them from spending any of their Bitcoin.



the orange is a block mined by Alice the orange block is one that just contains a transaction from Bob so here Bob is like some normal user, not a pool operator, they're not a miner they're just trying to put transactions up and normal blocks will just be those gray ones.

now let's say like what if Alice has the objective to censor Bob's bitcoin address and we're going to make the assumption that Alice knows Bob's bitcoin address and prevent any of their transactions from making it onto the chain, prevent them from spending any of their bitcoin. the first strategy is for Alice to just say none of my pools are going to include Bob's transactions what we call simple blacklisting, so would this work?

the answer is not all the way. Alice is over 51 likely to put up a block but this does not mean that Alice will be making every single block and there's still a possibility for other like miners, other pools not controlled by Alice, to include one of Bob's transactions in their block and thus get Bob's transactions on the blockchain. you're only going to be able to control every single block that appears on the chain if you own close to 100 of the compute so eventually Bob's transactions will be included in a block. at most with blacklisting Alice can only cause delays and inconveniences to Bob.

First strategy:

Alice tells her pools not to include Bob's transactions (**blacklisting**).

- Doesn't work unless you are 100% of the network
- Other miners will eventually include Brian's transactions in a block
- Can only cause delays and inconveniences

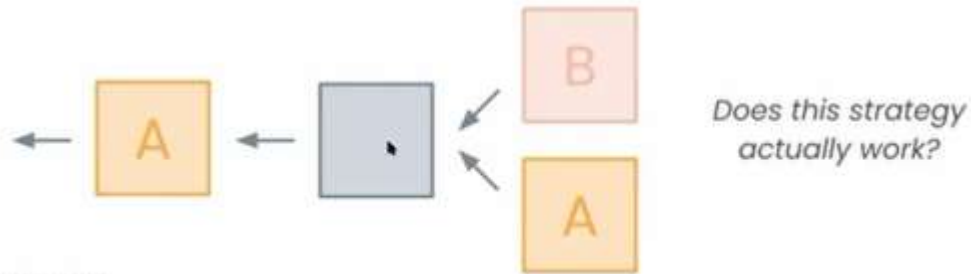


Brian should be Bob in the picture

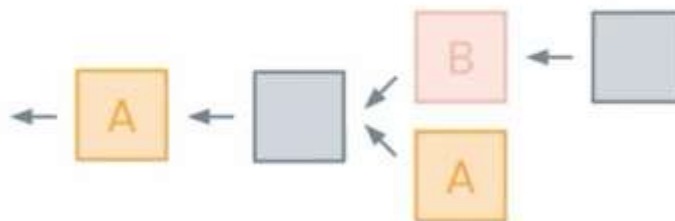
the next strategy is something that we call punitive forking. we should know that Alice's pools will not include any blocks that have transactions from Bob but also don't mine on top of any chain or don't mine on top of any block that might contain one of Bob's transactions and so what that means is that, let's say our chain is progressing here, Alice just put up a block whatever a normal block got put up on the chain and now let's say some miner that is not controlled by Alice goes and puts up a block and this block happens to have one of Bob's transactions in it at this point Alice would punitively fork she would force a fork by trying to upload their own block with a back pointer to not the block with Bob's transactions but the one before that right so that's how we establish a fork and that's the constraint that Alice's holding, I won't mine on top of any blocks that have Bob's transactions and remember that Alice controls over half of the hash rate which means that in time, you can just sort of think of law of large numbers, Alice is going to make more blocks than the rest of the network and so if Alice just sort of in a dedicated way keeps mining on her version of the chain she will end up making the longest chain and thus that will be the truth that'll be the ledger and this block with Bob's transactions will be omitted.

Second strategy:

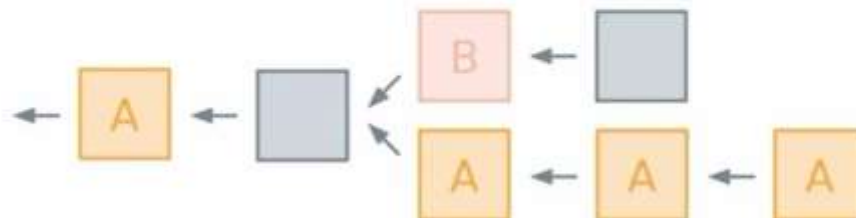
Alice tells her pools not to mine on top of any chain containing Bob's transactions, and announces this to the world.



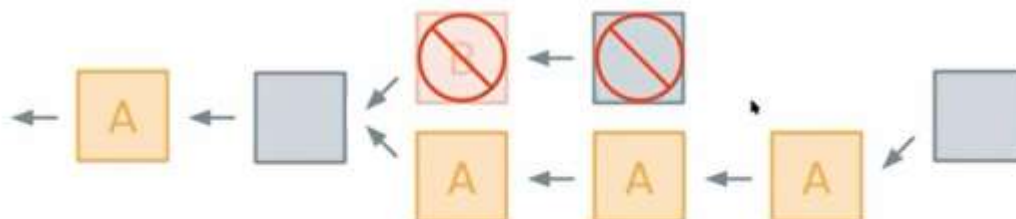
Remember, Alice controls >51% of hashrate...



So, in time, her pools will mine the longest chain...



Invalidating Bob's transactions, and those in subsequent blocks.



computative forking if you control over 51 of the hash power actually will work and as a result of this miners outside of Alice's pools they'll be aware of this, it's actually optimal for

Alice to announce this to the rest of the world and say other miners if you include bob's transactions in your block I will not mine on top of your chain and miners have to be aware of this and they will probably stop including Bob's transactions or they should stop including bob's transactions because again given that Alice has over 51 of the network these miners can rest assured that if Alice forks off that chain will be the longest one and that would invalidate their mining reward between all the work that they're doing so they want to be mining on top of Alice's chain the one that Alice considers true and they will also not include Bob's transactions and censor bob.

Miners outside of Alice's pools will stop including Bob's transactions when mining blocks, since Alice announced she would invalidate their blocks (and thus mining rewards)

Thus, we've shown that an entity with >51% of the network's hashrate can prevent someone from accessing their funds (or receiving any!)

this is a big like assumption that Alice would have more 51 of the hash power, is there other way ways to censor someone or try to censor someone without having this? there are and one of these ways is called feather forking

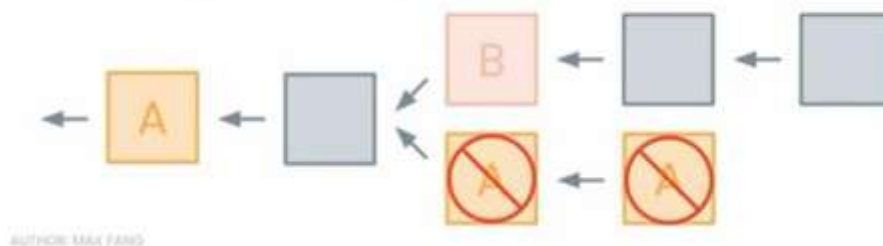
Feather Forking

Alice goes through the same process as in punitive forking even though there's less bite behind the bark so Alice again will announce that she'll fork any blocks that have transactions from Bob or she'll attempt to fork in the same way that she would have previously had. She had 51 of the hash power but you'll also say I'll mining on my fork but I'll give up and revert back to the version of the chain that has the block with Bob's transactions if that chain has some number k confirmation so basically at some point I decide that that other chain is far enough ahead of me that I just give up or that chain hasn't

enough work done on it that I just give up, so as opposed to in the previous situation Alice is committed to working on her chain forever because she knows that I have 51 of the hash power and my chain will be the longest in the end.

Punitive forking doesn't work unless Alice has $>51\%$ of hashpower. Is there another way? Yes! Called **Feather Forking**

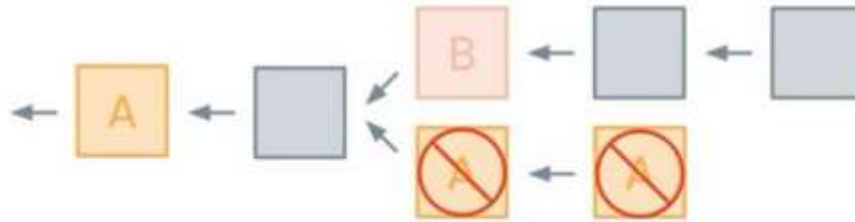
- New strategy: Alice announces that she will **attempt** to fork if she sees a block from Bob, but she will give up after a block with Bob's tx has k confirmations
 - As opposed to attempting to fork forever; doesn't work without $>51\%$



If that's no longer the case here so is it still worth it? puff up your chest and say oh I'm not gonna mine on top of any transactions or any blocks that have Bob's transit actions. well it might be worth it depending on how much hash power you control let's say Alice has some proportion q of the total mining power. in this case q between zero and one it's actually also less than one half because we're interested in the situation Alice has less than half and let's say that Alice will give up after one confirmation meaning that after Alice forks off if the fork that Alice is not working on, the one that does have the block with Bob's transactions, gets even one block added on top of it then Alice kind of throws up her hands and says all right I give up, we'll go with that version of the chain. that means if Alice wants to successfully censor Bob and orphan Bob out of there that means Alice needs to put up a longer chain before she hits her number of confirmations on the other chain and gives up. given that Alice is only going to wait around for one extra confirmation on that other chain that means Alice has to put up two blocks before one block gets put up on that existing version of the chain.

Punitive forking doesn't work unless Alice has >51% of hashpower. Is there another way? Yes! Called **Feather Forking**

- New strategy: Alice announces that she will **attempt** to fork if she sees a block from Bob, but she will give up after a block with Bob's tx has **k** confirmations
 - As opposed to attempting to fork forever; doesn't work without >51%

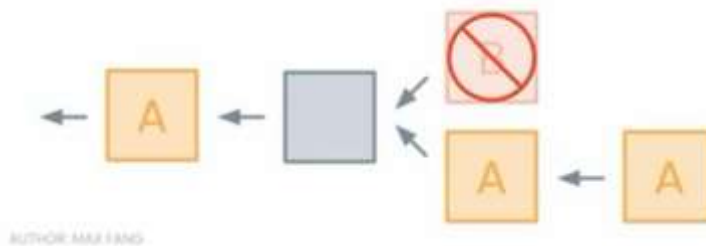


if alice has q of the network as a whole, that's her proportion of the hash rate, we again use that as the probability of Alice finding a block and if we want to say what's the probability that Alice finds the next block and the one after that it's just q times q or q squared so Alice has a q squared chance of successfully putting up the longest chain before she has to give up successfully orphaning Bob's block. let's say Alice has like 20 of the network's hash power of the network's hash rate then Alice under this strategy would only have like a four percent chance of orphaning the block.

Let q equal the proportion of mining power Alice has, $0 < q < 1$

Let $k = 1$: Alice will give up after 1 confirmation (one additional block)

- Chance of successfully orphaning (invalidating) the Bob block = q^2
 - Put up 2 blocks before 1 more gets put on top of the Bob block
- If $q = 0.2$, then $q^2 = 4\%$ chance of orphaning block. Not very good



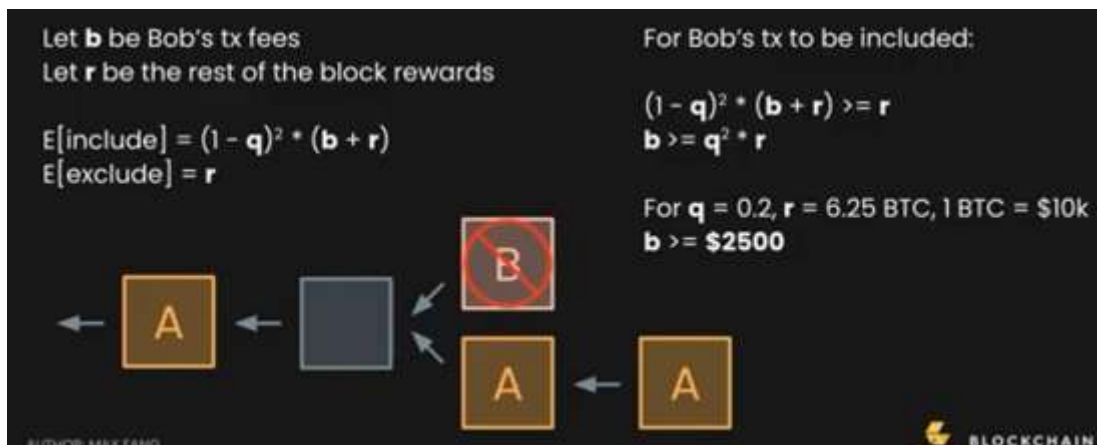
you would think is not very good but again the true sort of meat of the attack here is in the fact that Alice announced this to the rest of the world so now the rest of the world know that if they made that block with Bob's transaction in it it's got a q squared chance of being orphaned and so they got a way out like is it still worth it for me to include bob's transactions in my

block when I could just make a block without Bob's transactions and not be at risk of getting orphaned off by Alice.

the idea of the answer is Bob has to make it worth it in transaction fees for other miners to take on the risk of including his transactions so Bob needs to have a bit of money to incentivize miners to not censor him. let's say Bob can pay some amount b in transaction fees and should Bob not get put in a block then like the rest of the rewards that you would get from that block is r , let's say I'm a miner now and I choose to include Bob's transactions that means I've got a q squared chance of my block getting orphaned and a q squared chance of my expected value being zero because I would get no rewards, in the case that my block is orphaned which means I've got a one minus q squared chance of that block you know like staying up which is saying the rest of the network has to put up a block before Alice puts up two and then I would get Bob's rewards and the rewards from the block itself or I could just not play these games and just exclude bob's transaction and thus omit his transaction fees to me and just grab the r remaining rewards that that block would give me.

Anyway now there's no diminutive coefficient on that expected value because there's no risk of my block getting orphaned and so basically for Bob's transaction to be included greater than or equal to q squared times r .

for a 20 proportion of the hash rate there and current reward is 6.25 bitcoin this means that bob would have to pay like over 2500 in transaction fees just to make it worth it for his transactions to be included in a block.



Alice can put the financial pressure on Bob and effectively censor him this way, with this feather forking strategy.

Selfish mining

selfish mining explores a lot of how rational actors interact with the ecosystem and it helps us to understand a lot of game theory and game theoretical attacks. **Block withholding** imagine you're a miner who has just found a block and you decide to keep your block secret why would someone want to do that?

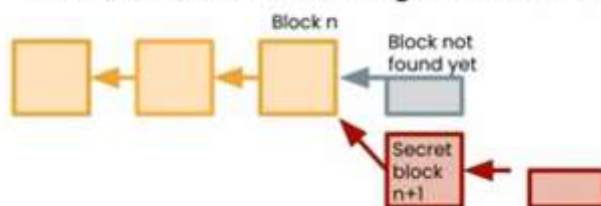
there are a few things or a few reasons why someone might want to do that, for starters by withholding a block you give yourself an advantage so instead of publishing a block to the network and receiving the reward, you can keep it secret because you haven't announced the block everyone is still looking for that same block that you just found so imagine you're participating in an easter egg hunt and there's one golden egg that everyone's trying to find and then that golden egg gives you a key to another easter egg hunt if you find the golden egg and everyone is still looking around for the same golden egg but if you haven't told people they're still looking for that golden egg when you've actually gone on to that next easter egg hunt and can find the next set of easter eggs and so because of this you get an inherent advantage and you can be

mining a step ahead of everyone else. when it comes down to like block withholding you want to be able to find the second block faster than anyone else can find this first one and because the way that people on blockchain networks validate, there's no truthful or honest chain, the way it will validate which chain is correct is they always mine on the longest one and at this time the network still thinks that it's mining on this longest proof-of-work chain when in fact you're mining on your own chain so you're at a slight advantage and you've actually created a longer chain and you've fooled the network and you've convinced them that pretty much whatever you want to broadcast these blocks then you can do this anytime and you get double the rewards because you don't only get the rewards from mining this block but if you achieve the second block you also get the rewards associated with this block as well.

You are a miner; suppose you have just found a block.

- Instead of announcing block to the network and receiving reward, keep it secret
- Try to find two blocks in a row before the network finds the next one

This is called **selfish mining** or **block-withholding**

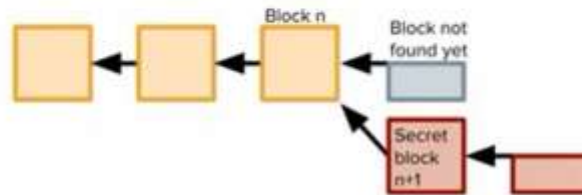


AUTHOR: MAX FANG

Note: "block-withholding" is also sometimes used in the context of mining pools - submitting shares but withholding valid blocks

If you succeed in finding a second block, you have fooled the network

- Network still believes it is mining on the longest proof of work chain
- You continue to mine on your own chain

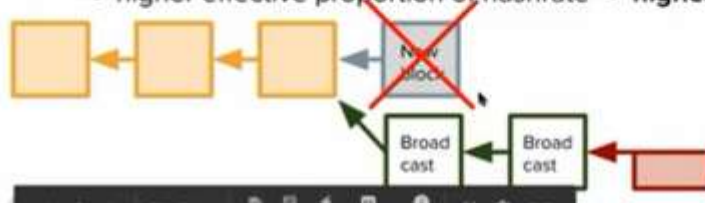


these have not been broadcasted yet but once you do broadcast them then all of the work that has been done by the rest of the network all to find this new block is now completely forgot about and pointless and because now once this is broadcast then now you're in charge and you've actually gotten the rewards from these two blocks so you actually get a little advantage on this new hypothetical easter egg hunt while everyone was looking for the initial golden egg and because of this quite simply you have higher expected profits, you're expected to earn more, if you can get these two blocks because you're the only one who is looking for these new blocks.

If the network finds a block, you broadcast your two secret blocks and make the network block invalid

- While network was working on the invalid block, you got a bunch of time to mine by yourself... for free!
- Free time mining on network

=> higher effective proportion of hashrate => **higher expected profits!**



what if your network found the new block before you could find the second one?

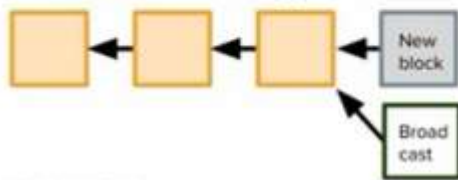
it's all about like getting traction regarding the block so you want to get enough people to fully start to mine on your block and if you have like over 33 of the binding power you can

actually lose this race of getting everyone involved every time but you still gain more profit and it all has to do with risk versus reward and depending on your mining power percentage you can still profit off of having this new block and convincing enough people to come work on it.

But what if the network found their new block before you could find a second one?

Race to propagate!

- If on average you manage to tell 50% of the network about your block first:
 - Malicious strategy is more profitable if you have $>25\%$ mining power
- If you have $>33\%$ mining power, **you can lose the race every time and malicious strategy is still more profitable!**



◦ (actual math omitted due to complexity)

there are a lot of complex calculations involved with it but essentially it just comes down to getting enough traction and enough people to propagate your block or your new pathway.

Selfish mining defenses

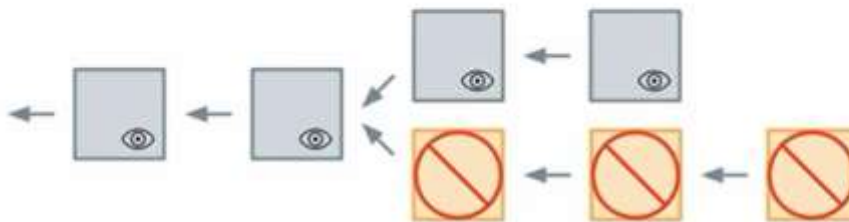
Certification of witness

we tried to include the original sources but it turns out that a number of these defense schemes were arrived at independently by different researchers on very different time offsets so you might find that it was not actually proposed by Schultz but someone else, but Schultz was one of the first people to propose this sort of scheme in 2015 which is where to ensure that every block is not being mined on in private selfishly, you just force that every block has to get a set of signatures from some random subset of other nodes. it goes like I'm a miner, I want to mine this block before I start iterating over the nonce I'm going to go ahead, I'm going to get some random subset of nodes that I have to reach out to and I'm

going to go ahead and send this block to all of them and make sure they all sign off of it and then I'll start computing proof of work and once I broadcast the block it's got those witnesses, those signatures on it, and this prevents selfish mining because let's say I made a block, I got the signatures and then I felt like found a valid nonce and I consider it complete and now if I want to start selfish mining on top of that block I still need signatures for this block and that's going to be selfishly mined on top of the previous one and if I am requesting signatures for this block and this block has a hash pointer to the previous one I sort of have to expose the fact that I had already fit down the block and broadcast that one as well otherwise the nodes that I'm reaching out to would not sign off.

Proposed by Schultz (2015)

- Accompany each block with a set of signatures from other nodes
 - Select a random subset of nodes
- Signifies that they were aware the block was being mined



AUTHOR: MAX FANG, APARNA KRISHNAN

this seems pretty great and it's honestly not even that heavy for the protocol, another set of signatures in the block is like a little bit beefy we want to be really efficient about block space but not bad because it does seem to effectively prevent selfish mining. you have to like tell the rest of the network about every block that you're starting work on but there are some issues with this as well so what this needs is a way to choose a random subset of nodes and a way to see which nodes were assigned to a given block so to basically verify that the right people did sign off on a block when I'm checking the signatures otherwise this scheme wouldn't be civil resistant (it means it

could be taken advantage of by someone creating multiple identities like multiple people that could sign off on the block) because otherwise if we're not just choosing random subsets of nodes in a really strongly random way you can imagine that an attacker would just make tons of identities and try to fit them or sort of optimize the selection scheme for how we actually select these nodes to ensure that it is their nodes that get picked in this random subset.

we want to make sure that's not possible and honestly the best way to do that is to make sure that the subset set of nodes is perfectly random and that way there's no way to optimize for it and of course we need to be able to check. if I'm taking a look at a block and I see some set of signatures on it how can I know that these are the right people that were meant to sign off on it and sort of to address both of these problems we need something called oracle. an oracle is some third-party service that provides off-chain data to the network which means anything that we're not tracking in the blockchain on the ledger itself (in bitcoin's case like anything that's not a UTXO) we need an oracle to ingest that information and include it in a block. we would need oracles for two things, we would need an oracle for selecting the random subset of nodes, if you come from a computer science background think API when you hear oracle it's a very similar thing, we need an oracle somewhere to query and say who are the random subset of nodes that I'm supposed to ask for signatures from and then later when I'm running consensus and verifying this block I go back to pretty much the same oracle, we might not need a separate one for this, and ask for this block is this actually the right set of signatures for it and is this the people that were meant to sign off on it? the issue with oracle's is that they're essentially a trusted point of failure like whatever oracle we're relying on to tell us these are the

nodes or these are the nodes that are meant to sign off on this block, that's sort of a point of trust where should that be compromised like now we can no longer have these assurances about no selfish mining.

the oracle problem is a very difficult one it's very useful in a bunch of different blockchains to be able to use external data in a way that you can verify that data but it's very difficult because it introduces this point of trust of what is the source, if it's not the protocol itself then we're putting trust somewhere.

Needs:

- Way to choose a random subset of nodes
- Way to see which nodes were assigned to a given block

Otherwise, this scheme is not **Sybil-resistant** (can be taken advantage of by an attacker creating multiple identities)

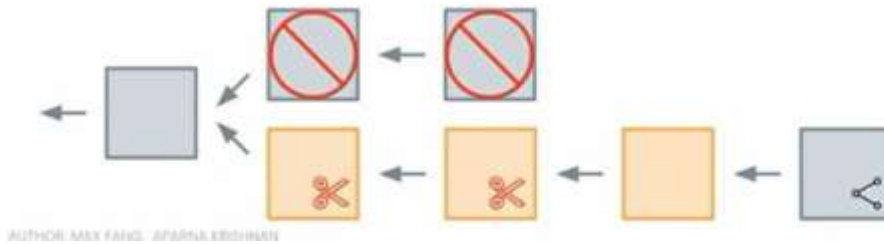
This requires a hard fork of the protocol, and relies on **oracles** (3rd-party services that provide external, or "off-chain," data to a blockchain network)

Fork punishment let's move on to another proposed defense called fork punishment.

let's use economic incentives and bitcoins fashion which is to say that should a fork occur, some other miner has an opportunity to prove that a fork occurred and we shouldn't get into the details of how you can prove that a fork occurred but it's not difficult essentially you just need the hashes of the blocks in the other fork, so if you incorporate a proof of forking in a later block then you get half of the total block rewards in that fork while the other half are destroyed, they don't go to the original miner.

Proposed by Lear Bahack (2013)

- New approach: use economic incentives
- The first miner to incorporate proof of a fork in their block gets half of the total block rewards in that fork, while the other half are destroyed



if we take a look at the diagram above let's say the honest network is going and we end up at this left most block and then let's say the network makes these two gray blocks(those with a forbidden sign on it) but the network know that all the while there's some selfish miner who's been working on those three orange blocks and the other miners they put up those two great blocks and then out of nowhere the selfish miners are like all right time for me to take control of the chain and put up the longest one and then the miner releases these three blocks (the three orange ones) so now that's the longest chain that's the true ledger um but there were those miners who spent all that work on those other two blocks that just got orphaned off in the fourth so what some minor can do then is in the next block (in that last the rightmost gray one) they'll include a proof of forking so they'll basically include like references to those two gray blocks there and show you that those two blocks were overwritten (the blocks with the scissors) by a fork here's the proof and then the miner that puts up that last block the right most one will get half of the rewards from the two scissor icon blocks and the original miner of those two blocks doesn't get any rewards, they're burnt.

Drawbacks

- Honest miners are caught in the crossfire
- Makes fundamental changes to Bitcoin's monetary policy
- Also requires a hard fork
- Doesn't completely prevent selfish mining
 - If an attacker publishes a selfishly-mined fork that is $>1.5x$ as long as the competing one, they'll still be profitable

drawbacks that make big changes to the bitcoin's monetary policy, this sort of constant that we have hold that we're gonna hit this like 20 million hard cap but now we're saying no we're introducing a mechanism for bitcoin to essentially be burnt permanently that half of the reward that no longer goes to the original miner doesn't make its way back onto the chain later, so you can imagine a lot of people were unhappy about this they're like no I like a predictable supply in bitcoin and there's a lots of different derivative arguments from not only the shape of like the supply curve is it predictable or not, is it finite, there's so much that can stem out of that so that caused tons of discussion and there's also issues like before that even honest minors end up getting caught in the crossfire because sort of benevolent or accidental forks can happen. it might just be the case that two miners unaware of each other were mining found the block at the same time broadcast it at the same time and have caused a fork because some miners that got the first block now have to choose is the second block the one that I should start mining on top of or not so these natural races can occur and honest miners can find themselves being cited in a proof of a fork and find themselves losing rewards because they accidentally put up a block that forks another when they thought they were just mining honestly. that's not great and this requires a hard fork, when we're talking about a hard fork that means you should decide not to update your version of bitcoin like your blocks would not be

considered valid in the sort of new updated version and that's because there's a lot of protocol changes that we have to implement here for example bitcoin being able to be burnt and proofs of fork being able to be included in blocks in general and this won't even completely prevent selfish mining it just kind of puts expected loss on it or it adds some loss to it that you have to outweigh. if an attacker publishes a sufficiently long fork that's like one and a half times as long as the one that they're competing with they'll still be profitable. that's like a pretty absurd amount of computational power that you would need but it's not a perfect solution and it requires a hard fork, by the way the previous one that we were talking about the certification of witness also requires a hard fork reason being not only because it relies on oracles and we need a way to interface with them but also because it now requires that blocks include the certification of witness piece that older versions of the client just wouldn't use. we tend to hate hard forks um because it always divides the community, the sort of user base, into the people who support it and who don't and now also in fork punishment we have these additional drawbacks that it's not a great solution it puts honest finders in the crossfire and there's fundamental changes to bitcoin's monetary policy.

Uniform tie-breaking

Proposed by Eyal and Sirer (2014)

- In the event of a tie, miners **choose randomly** which chain to mine on
 - Attackers can't benefit from superior network connectivity
- Attackers would need to command **$\geq 25\%$** of network hashrate for selfish mining to be more profitable than honest mining
 - Right now, selfish mining is better even for miners with **0%** of the hashrate, if they're optimally connected
 - Selfish mining will **always** be better for miners with **$\geq 33\%$** of hashrate

the last defense for selfish mining that we'll talk about is something called uniform tie-breaking it's a pretty sort of like naive straightforward approach but it ain't bad. this is one of those papers that was proposed by a couple people in very close time frames. when you spot forks it's not like I see oh a fork with three blocks on top of it and one with two blocks on top of it, it's usually like just uh one block competing with one other block, not that that's really relevant but that's just kind of helpful to keep in mind sometimes.

in the event of a block tie or a fork, miners would choose randomly which side of the fork they want to continue mining on and what this means that attackers can't benefit from being highly connected from getting their blocks out faster because that's really helpful in selfish mining the strat is you have your block, it's ready, you wait for someone else to broadcast theirs and you try to beat them to the punch so that relies on you having good superior network connectivity, you're being able to beat them to the punch, but now we're saying look it's a 50 50 split and this is sort of like we are reducing it to the case where we're only expecting a two-way fork. now we're basically boiling it down to instead of it's whichever block I saw first that's the one I'm mining on top of, which is sort of the go-to implementation for the bitcoin protocol which is also why well-connected clients or optimally connected miners they perform really well because the default implementation for

miners is to say whichever one I saw first I'm gonna start mining on top of it. now we're switching it to instead of whichever of the two blocks I saw first online on top of that one now it's a 50 50 random choice and this means attackers would need to control at least a quarter of the network hash rate for selfish mining to be more profitable than honest mining forcing attackers need to have some minimum threshold of the computational power for selfish mining for the expected value to be greater than that of honest mining so right now like selfish mining is a higher expected value even for miners who basically control no amount of the hash rate if they're optimally connected like if they know that they can get their blocks out before or they can get their blocks picked up by other miners before everyone else then it's always worth it for you to selfishly mine because let's say it's my lucky day, I just found a valid block and I know that I can get this block to the network faster than anyone else, there's no good reason for me not to mine on top of that block because I can rest assured that even if someone puts up a competing block right now I'll get my block out faster and it will be part of the longest chain so now we're like taking that out, we're saying look even if you can get your block out faster until another block is mined on top of either your block or the competing one miners are going to be going through this 50 50 choice of whether or not to mine on top of yours which means you're hoping that enough of the rest of the network sees your block and chooses your block in that 50 50 choice for the block that you were selfishly mining on top of that to really even be considered valid. selfish mining is always better for miners and by always we mean regardless of how well connected you are, if they have at least 33 of the hash rate, we're saying that even if you've got the worst possible

connection you're gonna need at least 33 of the hash rate for this to be a profitable strat for you.

miners when they see a fork they choose randomly which side of the fork to mine on and that makes it a lot riskier to selfish mine because the odds might not be in your favor and the rest of the network might choose to mine on not your side of the fork.

Lecture 6: Trust without trust

<https://www.youtube.com/watch?v=KhBNWlQHwFg&list=PLLkIRvMHnp6OIWkZVa85g2tv7NtlgoAID&index=6>

Distributed systems

what's the purpose of studying consensus? why does consensus matter? for starters consensus is just having a general agreement and having a majority of everyone involved to agree on something we see, like in the united states voting system when you vote for presidents or different people for office you want to define the majority of US citizens to be able to vote and elect the next leader and just like we have that in our government system we also have consensus within blockchain with across multiple devices, let's say we have person A here and person A wants to get pizza for lunch so person A reached out to person B and C and asks them what they want for lunch and they say they want burgers but that being said we all want to go to get lunch together so how do we come to an agreement on what we actually end up getting for lunch? what we can do is all interact with each other and we can see that the origins of consensus (finding a majority agreement)comes from planes like digital avionics, this comes down to the fact that when planes were first invented there were a lot of different subcomponents that all handle specific parts of a flight like a classic flight there's a lot more communication that happens across all the different devices, they measure altitude, velocity, thrust and all these different components of planes that are really crucial towards having a safe flight. in addition to

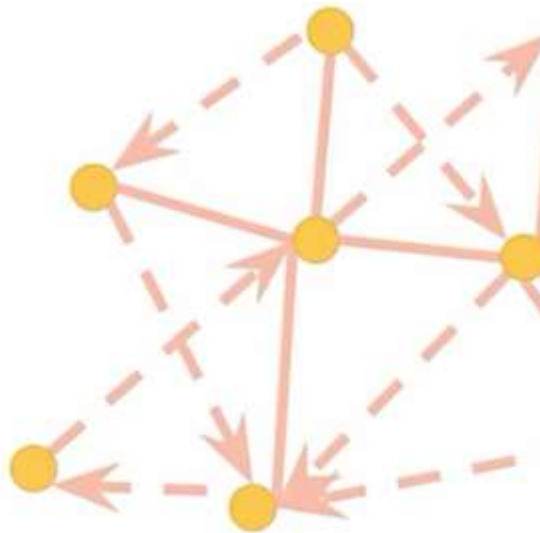
caring about the passenger safety which is really huge it's important that to note that aircrafts cost a lot of money and any possible damages could be really problematic to the company itself.

with all these different components the real question is how can we get them all to talk to each other and be synchronized in an effective way? the solution to this is through creating consensus among these devices. there are a few problems in distributed computing and a lot of them reduced to consensus. the million dollar question is how can we create a reliable system from unreliable parts (systems that are all working and all have minds of their own how can we connect them in a way where we can really trust with pretty much 100 certainty that they will execute the tasks that we're interested in and that we care about)? like multiple machines working together we see them in space launches and self-driving cars or in database centers all interacting with each other and of course in blockchain. within all these systems it's really important that we can trust that all these devices will work together and work synchronously with or work in unison without being able to trust the reliability of each component. this comes down to the idea of what we often say is trust without the trust, we can trust the correct execution of the distributed system so that all the different devices will work together effectively towards a common goal but we don't need to trust any individual component and in case we do have issues or one of them breaks down malfunctions or something happens we know that we can trust that the greater goal will be executed now we'll just get into a few definitions:

Node: node is just a device and so all these dots here represent different nodes they can be computers different computer programs or overall just devices.

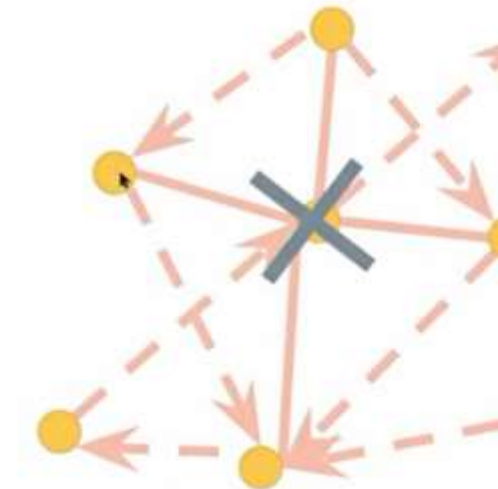


all these nodes interact with each other so they like they pass on information from one node to another they take in information and they all work together to accomplish a common goal.

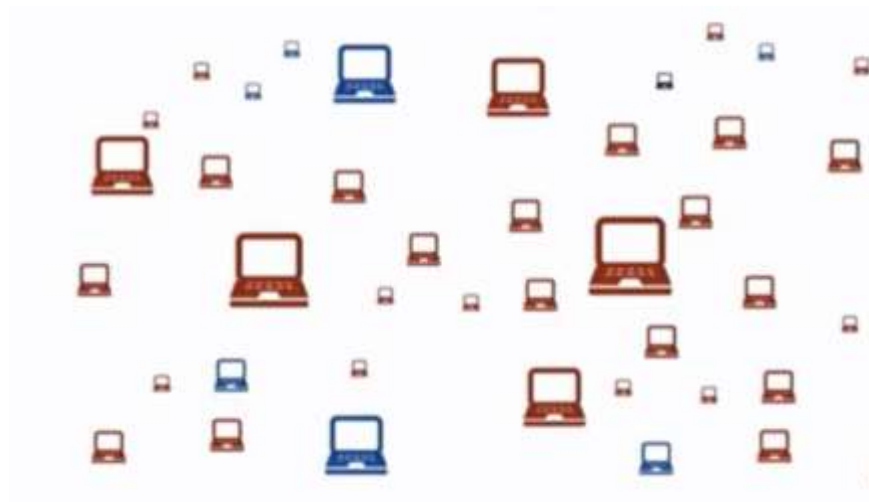


one of the challenges is having with all these different distributed system and with all these different nodes is that there is no global clock that's synchronizing all of them and it's really difficult to line everything up in a way that makes them effective and able to work reliably together. in case of a failure of any individual component here that can be really

problematic if you're relying on it for a key component of your common goal and there's no global dent like NASA world clock that's synchronizing everything too. working with all these moving parts can be really difficult. if one of the nodes fails then we lose a lot of communication here so how do we manage this?



one of the initial ideas of consensus came from this idea of binary consensus (the reason it's called binary consensus is because all the nodes randomly output a number whether it be zero or one just like in binary) as a way to get everyone on the same page and get all the nodes to come to agreement with each other.



all these computers output their zero or one then they the nodes as a general all communicate with each other and find out which is the minority whether it be red or blue, zero or one and so once they've found whatever is the minority they're all able to converge to a single value of in this case one because red was the majority over all the devices.



the big idea is that distributed systems really work to find a way to do meaningful work while there is chaos where there is uncertainty and maybe we have devices that are down nothing is synchronized and just finding a way to work effectively with all these different barriers that we face.

Properties and cap theory

safety and liveness a set of conditions these both describe things that the system must accomplish and things that must not happen.

Safety: safety is like the set of things that will not happen in a distributed system and to give you an idea here an is you won't lose all of your files if the google drive site goes down so that's a constraint that your data is safe in this way.

Liveness: similarly we have liveness constraints which are things that must happen like google drive will successfully fetch your file every time you click on its icon in your google designing a robust distributed system is really the issue of balancing these different constraints safety and liveness I want

it to be responsive enough so live enough that it's a good user experience and you can interact with the system quickly get the data you want or maybe write in the data that you need but I want this system to be sufficiently safe so if multiple people are interacting with it at once they won't trip over each other's toes and mess something up in the system if nodes go down like we mentioned am I just gonna lose everything on it when it comes more specifically to consensus which really is like the task of the system agreeing on either some execution (some operation) or agreeing on some shared value the system has to update or maintain some shared state and agree on how to move forward with this and how we consider consensus to be correct, it isn't necessarily rigid, you as the designer of the system have to figure out what does correctness mean if we're looking for a majority vote let's say our nodes can output a zero one or a two now it's possible that we don't have a majority in any one of those outputs is correct consensus to say we abstain on this voting round and there's no winner.

So you have to define what correctness is and the point of that. there's sort of three criteria on along which you might define the correctness of consensus:

- validity: any value decided upon must be proposed by one of the nodes (let's assume that the task of consensus is to as a group decide on some value (like the value being zero or one or what are we gonna eat for lunch) so this value must be valid, it must have been proposed by any one of the nodes participating in consensus).

- Agreement: All non-faulty nodes must agree on the same value (the final value must be agreed upon by all non-faulty nodes so every node that is successfully participating consensus must be unanimous agreement that they all have the same output from it).

-Termination: All non-faulty nodes eventually decide (all nonfaulty nodes will eventually reach some value of their own we can't allow consensus to block or to hang on some node that's taking forever with making a decision). what is the optimal consensus algorithm and what are maybe the factors of consensus that we might want to optimize across?

The answer comes from a theorem called the cap theorem which first says that the three areas that you'll want to maximize across a distributed application are:

- consistency: every node provides the most recent state, or does not provide a state at all(which means that once some node says this is the current state of the system (this is what consensus like most recently said) every node must provide the same value for this or not provide any value at all so that means like if I query any node in the system and say what was the last consensus value it should not be possible for that node to give me the wrong answer that's consistency).

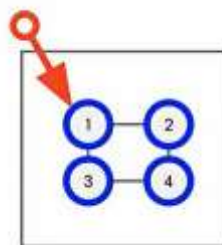
- availability: Every node has constant read and write access (this is sort of like the responsiveness of your system, every node should have constant read and write, if my system is like very available that means that every time I ask a node what's the current state (what was the last consensus value) it should be able to tell me and it shouldn't be able to say anything like I'm disconnected from the other nodes right now so I'm not sure or if you want to say can you update the state of the system to read blue as my favorite color instead of red the node should be able to go and do that right for you).

- partition tolerance: The system works despite partitions in the network (which means your system will work and work is deliberately flexible here it's your job to decide what correct consensus is but you should be able to achieve correct consensus in the case of arbitrary what we call partitions in the

network and a partition is really just a sort of sets of nodes that can't talk to one another so we have to be able to deal with sort of lapses in communication nodes going offline or being disconnected from one another, the system still needs to be sufficiently interactive and achieve its desired goal) the cap theorem essentially states that you can't really have an optimal distributed system across all properties, you can't have a system that's perfectly consistent available and partition tolerant and you sort of have to compromise, the more of any one of these properties you try to build towards the more you have to sacrifice in the other two.

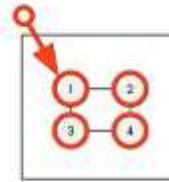


proof for why all three are impossible:
let's take a look at this example here

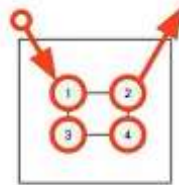


we have a distributed system with four nodes and although we haven't drawn out all the connections here so all of these nodes are connected to one another and the goal of the system is essentially binary consensus it just stores, it's like a little distributed data storage that stores what is my favorite color, so right now the shared state of the system is blue and we're about to start off with a write query basically to node one saying node one can you please update my state of the system

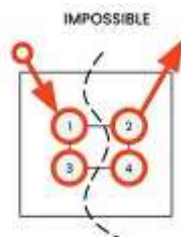
to be red and because the system is available it does have right access so it will successfully complete this action for you and also since the system is consistent every single node will adopt this new value.



by consistency and availability here asking node one to do this update has turned all nodes to keeping the value red as what they see as the state of the system, so now if I was to like ask node 2 what's my favorite color it would tell me red



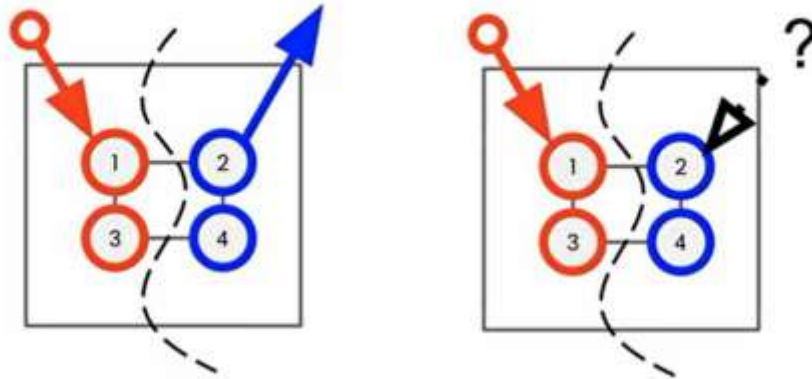
and this is again demonstrating consistency like it's giving a consistent answer with my last request to update the state of the system but now let's say we were to drive a partition into the network so previously every node was connected to every other node that's why there was no issues spreading the value that my favorite color is red to the three remaining nodes but now we're saying nodes one and three can talk to one another and nodes two and four can talk to one another but one can't talk to two or four and three can't talk to two or four like you see the partition there.



now we'll show that we can't actually maintain both consistency and availability assuming that partition tolerance is

also here, we're assuming that the system can have all three properties and we're gonna show that by making that assumption one of the three properties must break so if we're assuming that the system is partition tolerant then it means that if I was to try to go for this example here

Partition Tolerant + Available = Not Consistent Partition Tolerant + Consistent = Not Available



if I was to for example go to node 2 and say let's turn my favorite color back to blue by the availability guarantee node two is gonna go and successfully try to execute that for me so it'll try and update the state to my favorite color which is blue and so maybe node four will catch window that because it's connected and it's not partitioned but nodes one and three wouldn't get that update because there is like a network partition there, and if I was to ask node one what's my favorite color it would say it's red and thus like right now we just violated consistency by assuming that it was partition tolerant which is saying it would continue functioning even though it's partitioned and by assuming that it continued to be available so it would actually go and execute this like update my favorite color to be blue we've shown that it ends up breaking consistency and so this like you can try the different sort of like sets of assumptions here like what if it stayed consistent while it was partition tolerant you'd show that it would have to be

unavailable it could have to say no I can't update your color for you so basically you can't have all three but when we're talking about distributed systems in the real world like the internet for example or a series of redundant computers on an airplane we kind of need to be partition tolerant we need to be able to get valid consensus results from the system even if all the nodes aren't successfully connected to one another because you're not really able to assume that, you can reach anyone else on the internet at any given time so we have to be sufficiently partition tolerant as a system when it comes down to designing these things it's really about a choice and not a black and white choice between I'm going to have a completely available system or a completely consistent one but more like how much am I going to sacrifice in the name of consistency to have a more available system and it's like deciding on those trade-offs and that can lead to some really interesting different implementations you can do like something that some database systems do is called eventual consistency where if you're talking to one of the servers and it's disconnected from the other ones it'll still let you write in new values to the database and at some point in time when it syncs up with the other servers it'll like merge all your changes so that's like one cool trade-off where technically your system will be inconsistent for a little bit but you've got this eventual consistency so you don't have to worry too much about that the high level idea here is that you can't have a system that has all three of these properties.

Voting-based consensus Paxos:

the way paxonian parlame would implement their own government structure was through three main roles they'd have a **proposer** was like a legislator who advocated a citizen's request and then moved the protocol forward the **acceptor**,

acceptor was a legislator who mostly voted and so it's the person who had a lot of the say in the future of the government and then there was the **learner**, learner is was held responsible for remembering and carrying out results for the citizens.

quorum within the pakistani parliament was any majority of acceptors and any quorums must overlap (there must relates idea of majority and that majority must be constant and that majority can't change too much). the main idea is the protocol happens over a few rounds and each round is like a piece of legislation that one needs to be passed or something that people are voting on and each piece of legislation has these two phases a preparation phase and an acceptance phase. preparation phase for starters the citizens talk to proposer and the proposer is the one who proposes a new idea up to the entire parliament and then within the parliament the proposer acceptor and learner discuss. what happens in this phase is that they assign a decree (which has a number and a value) so they say for decree number 277 the sale of brown goats is permitted or all these different examples in the picture below demonstrate a connection between a value and then whatever the actual proposition is and then if quorum is achieved if there is a majority an agreement for or that is a majority that is in agreement and is for this proposition then the decree passes and then the process repeats after the learner talks to the citizens.

1. Citizens talk to Proposer
2. Within Paxos parliament: Proposer, Acceptor, and Learner discuss
 - a. A decree has a number and value
 - b. Decree passes
3. Learner talks to citizens

1375: Γλυδα is the new cheese inspector

277: The sale of brown goats is permitted

37: Painting on temple walls is forbidden

we can draw this connection of the pakistani parliament to our world of blockchain and so the correspondence is sort of as we see in the picture below:

<u>Parliament</u>		<u>Distributed Database</u>
legislator	↔	server
citizen	↔	client program
current law	↔	database state

legislators we can see that even within a distributed database the legislators have the role of a server so the servers are the one who have a lot of the say and orchestrate a lot of what we see within distributed systems whereas we have citizens which relate to the idea of client programs and then ultimately at the end of the day whatever is passed or failed that current law relates to the state of the database and what has been approved by the database and what is currently in action. this is an algorithm that's used by a lot of really large companies and is pretty widespread, a few reasons for this is that it is effective, it has good performance and it is a real pretty quick process and it can be used with really significant sets of data as well so terms of replicating and manipulating set of data and it assumes that there are no failures or byzantine failures.

Raft:

raft is another consensus mechanism that is designed to be simpler like a simplified version and of paxos and it has a leader-based approach and is pretty effective to implement. A raft cluster (we think of little groups of nodes as clusters) has one elected leader so this is one member who's in charge of the following responsibilities:

- 1.the leader communicates with the client directly and respond
- 2.is responsible for managing log replication on the other

servers of the cluster (what this means is this leader is the one doing a lot of the communication and is the one who actually is authorized to distribute logs or list of transactions or whatever it may be any information that is sought to be held on a raft network).

3. the leader is the leader until it fails or disconnects and the way number 3 happens connects to this idea of virtual heartbeats or little pings that are sent out like vibe checks that are sent between the leader and the rest of the network so the leader sends out these heartbeats to other nodes and it indicates that the leader is still aligned still, connected and still executing its responsibilities as a leader, if any of the other nodes no longer receive the heartbeat or the little messages that are sent then they have a randomized internal timer that automatically starts running and the reason why this is randomized is because it ensures that the whole process of electing the new leader is also randomized so when the first person or the first node doesn't receive a heartbeat then a randomized internal timer starts and then the first candidate who has their internal random timer complete or timeout then they become the new leader.

Now let's talk about log replication (how the leader is responsible for managing log replication on the other servers of the cluster) log replication is done through by a leader accepting and ensuring that these client requests go through and that all other nodes have also followed the request so the leader is responsible for ensuring that other nodes comply and work with the leader. Here is a review:

HOW IT WORKS

A Raft cluster has one and only one elected **leader**

- Communicates with client directly
- Responsible for managing log replication on the other servers of the cluster
- Leads until it fails or disconnects, in which case a new leader is elected

Leader Election

1. Leader sends "heartbeats" to other nodes saying that it is online
2. If other nodes no longer receive "heartbeat," they start an election cycle (a random internal timer)
3. First candidate to timeout becomes new leader

Log Replication

1. Leader accepts client request
2. Leader ensures all other nodes have followed that request

Byzantine fault tolerance

in all of our distributed systems so far we've sort of allowed nodes to malfunction (maybe become disconnected or time out or have to go offline and online again) but we haven't really allowed or sort of factored in the possibility of nodes deliberately acting in a way that throws off consensus and either tries to point it towards a different value or just make consensus unachievable in the first place this behavior is something that we call byzantine behavior and it comes from this paper called the byzantine generals problem.

The Byzantine Generals Problem

LESLIE LAMPORT, ROBERT SHOSTAK, and MARSHALL PEASE
SRI International

Reliable computer systems must handle malfunctioning components that give conflicting information to different parts of the system. This situation can be expressed abstractly in terms of a group of generals of the Byzantine army camped with their troops around an enemy city. Communicating only by messenger, the generals must agree upon a common battle plan. However, one or more of them

which is sort of explores um like how one might deal with malicious nodes through analogy of a set of generals attacking its city.

there's a general and their lieutenants are planning to attack a city so everyone's sort of waiting around on the general to give a specific order like okay we're either going to attack right now or we're going to retreat, but no one can really trust anyone

here so any given lieutenant can't trust necessarily any other given lieutenant or the general (the other lieutenants might try to spread fake orders that they claim they have heard or just not respond so the same sort of sets of faults that we've been considering timing out going offline or just lying and being malicious) so pretty much everyone's waiting around on the general to issue an order and there's some scheme by which the lieutenants will share the order amongst themselves in an effort to reach honest consensus and here our consensus is correct if it satisfies two conditions; the first is that all the loyal lieutenants will obey the same order and the second is that if the general is loyal, all the loyal lieutenants will obey the order that was sent to them by the general (if the general is loyal then all the loyal lieutenants are going to do what the general says) there's actually no like efficient scheme no real way to do this if there's at least a third of the network being controlled by traders and it more or less boils down to just the scenario that's pictured below:

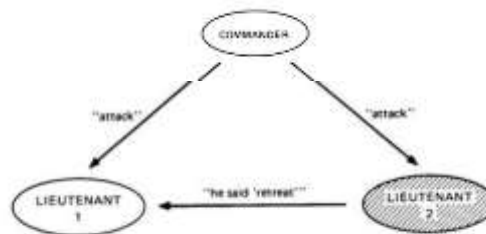


Fig. 1. Lieutenant 2 a traitor.



Fig. 2. The commander a traitor.

if you're lieutenant one then let's take a look at the first scenario where we've got an honest general or commander and

a dishonest other lieutenant so I'm receiving an order from the general saying it's time to attack but this dishonest lieutenant two is telling me no he said retreat even though lieutenant two also got the order to attack so okay I'm getting attacked from the general and retreat from the lieutenant not really sure what to do but you can't just sort of feel safe and assume okay let me always attack or let me always retreat in this situation because if we take a look at the sort of the flip side of it where we've got a dishonest general and an honest lieutenant, from lieutenant one's point of view they're in the same situation it's still the general saying attack and lieutenant two saying retreat and that's all the inputs that lieutenant one really has available to them so you can see that in both of these two situations lieutenants are kind of looking at the same picture but they can't tell if it's the commander or the lieutenant that's being dishonest so you can't really know which order to obey that'll necessarily satisfy those two conditions. we're not going to be able to achieve a consensus such that again all loyal generals say the same thing and if the general is loyal they do what he says, it's not gonna be possible if there's at least a third of the network being traitors so this is an interesting constraint to be fighting against because obviously within the realm of bitcoin where it's not just me running my own data servers I'm not making sure that all of my machines are not running malicious code like anyone in the world can run their own bitcoin node so we have to be resilient to this sort of byzantine behavior like someone might write their own version of the bitcoin client and start trying to send like faulty blocks or whatever so we have to fight against this constraint somehow but first to draw the parallels between the byzantine general problem and a blockchain network we have:

Byzantine Generals

Generals

Traitor generals

Geographic distance

Attack or retreat

The Blockchain

Nodes

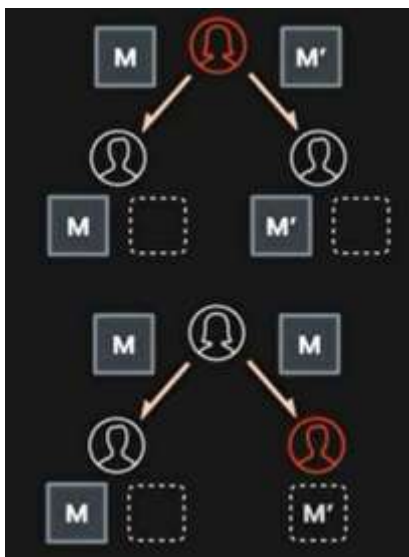
Faulty or malicious
(Byzantine) nodes

Distributed network

Consensus on history
(e.g. transaction log)

if you're trying to disambiguate a bit between generals and lieutenants you can think of this problem as being the same thing if it was a group of all generals where the difference there is that the general is the one kicking off the message so you're saying that anyone might issue the order to attack or retreat as opposed to one person and that's kind of how it is in the blockchain network, any minor can suggest a block we're not just waiting on one person to do it so generals are more or less nodes we've got traitor generals or byzantine generals which we would also call byzantine nodes these are just like malicious miners or something like that. the notion of geographic distance in the byzantine general's problem, the reason that they have to pass messages to one another and can't just talk in a group, is the same as the notion of our network being distributed, we have to pass messages across the internet one to one and the decision to attack or retreat is the same thing as saying do we or do we not add this block or what is the next block? you can view proof of work as a sort of probabilistic solution to the byzantine general's problem so it'll like let you achieve consensus in what we'll call *the three general problem* where we showed that if one of three generals is dishonest we can't do it basically the issue with the previous setup is that there's no way of knowing what lieutenant is going to

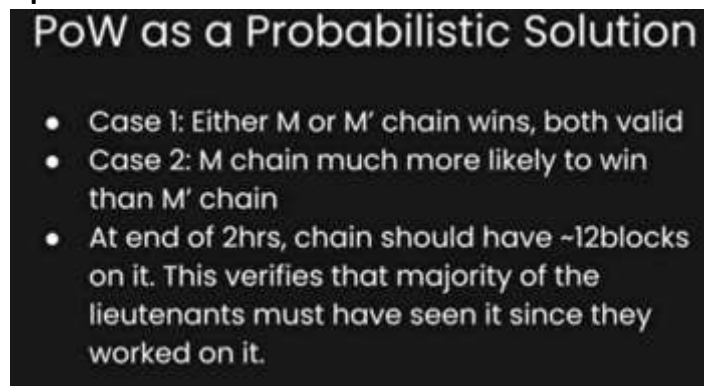
acknowledge the result, if I'm lieutenant one here I know that lieutenant two told me that the attack was retreat but I can't really know that they'll commit to it on their so our goal here is to know that every other lieutenant saw the same decision that we saw and is gonna end up choosing it, what we do is we basically have everyone solve proof of work whenever they want to share a message and we have them follow this little scheme where the general will decide on an order solve proof of work for it like take the order attack or retreat put it in a block, run proof of work and then when they've solved proof of work successfully they'll broadcast this to all the other lieutenants and now let's say you're a lieutenant whenever you receive a broadcast it'll be like some chain some number of blocks linked together and if that chain is longer than the one that you're currently storing on your own end then you'll overwrite your chain with that chain and then you'll try to solve proof of work for another block on top of it and when you succeed in doing so you rebroadcast and everyone then agrees that. at the end of two hours whatever my chain that I'm maintaining is, I'll be the order at the first block of it. let's see how this play out



this is the same situation from the lieutenant's so let's take a look at the first case that lieutenant Juan would have to disambiguate from which is where the general is being dishonest which means that what the general would try to do is send you know one message or order to one general and a different message to a different general like attack to one general retreat to the other, what would end up happening though is the two honest generals would both receive this first block and they would go ahead and start doing proof of work it really would be like the general would also start doing proof of work probably on either one of these two on m or m' so whichever of those two blocks the general also decides to do proof of work on m and then a block on top of it first or lieutenant two could broadcast m' with the block on top of it. first it wouldn't really matter the point is whichever one gets broadcast first the other honest lieutenant would see that and in classic like bitcoin proof of work style they'd abandon their shorter chains start working on this new long one and that means that chain with m at the root one now lieutenant two is also gonna adopt that chain and start mining on top of like and at the end of the two hours even though the general tried to get some lieutenants to think the order was m and some lieutenants to think the order was m' they all end up with the same longest chain and they'll do either one of those two options whichever one they choose would actually still satisfy those conditions for correctness so that case is trivially solved but in the second case this is where we show how it's only like a probabilistic solution in the second case where we've got like an honest general saying like attack or retreat but a dishonest lieutenant that is disseminating the orders, what would happen is that the general would send the same message(the same root block to both of the two generals) in the picture and the

general and the honest lieutenant would start mining on top of this message and trying to come up with the next block and if the dishonest lieutenant wants the network to believe that my order m prime is the actual real one this dishonest lieutenant would have to mine two blocks the root with m prime and one block on top of it before the honest lieutenant and the general so the rest of the honest network combined before they could come up with just one block so you're much more likely to end up with the honest chain the one with the non-forged message being the winning one but there is still some probability that the dishonest lieutenant can get ahead early in the race but proof of work works such that if the whole network is working on the same block it should take them about 10 minutes to come up with this block if at the end of two hours our chain has about 12 blocks that's about 10 minutes per block then we know that it took about the entirety of the network's computational power to solve proof of work many times in two hours and so if we see a chain that's long enough that means we can rest assured that a majority of the network had to work on it to make it that long within a long time and since the majority of the network worked on that chain, we know that must have been broadcast to them so that they could have been computing proof of work hashes on top of it so using proof of work we have this fairly strong probability that the correct answer wins out along with the assurance that the other honest lieutenants will end up choosing the same thing as you because they'll have the same chain and this also gets the results become stronger when you include the possibility to sign these messages which means you'd be able to tell if it was tampered with or didn't actually come from the general and there's a non-proof-of-work or non-fancy probabilistic solution to this using signed messages in the original paper and the issue

with it was that it required nodes to be sufficiently well connected, it required that any two honest lieutenants could speak to each other only talking through some number of intermediary nodes and in the blockchain network we can't maintain the sort of guarantee that you'll be sufficiently well connected to every other honest bitcoin node so the reason we have to adopt this probabilistic solution to the byzantine generals problem is that we have to be sufficiently partition tolerant. the idea here is that while we maintain that it's impossible to definitely achieve consensus when there is at least one third of your network run by traders we can come up with clever sort of probabilistic solutions like proof of work that combined with some conditions about the bitcoin network really allow us to essentially bypass the byzantine generals problem. So a quick review:



PoW as a Probabilistic Solution

- Case 1: Either M or M' chain wins, both valid
- Case 2: M chain much more likely to win than M' chain
- At end of 2hrs, chain should have ~12 blocks on it. This verifies that majority of the lieutenants must have seen it since they worked on it.

Nakamoto consensus

what we just revealed in this previous section was that the consensus algorithms that we had mentioned for example paxos and raft were not assuming to operate in a byzantine environment, they weren't placing themselves under the possibilities of nodes acting maliciously so they didn't have to worry about solving the byzantine generals problem or trying to solve it but now let's talk about a series of consensus algorithms that have this shared approach to solving the byzantine generals problem where they require you to commit

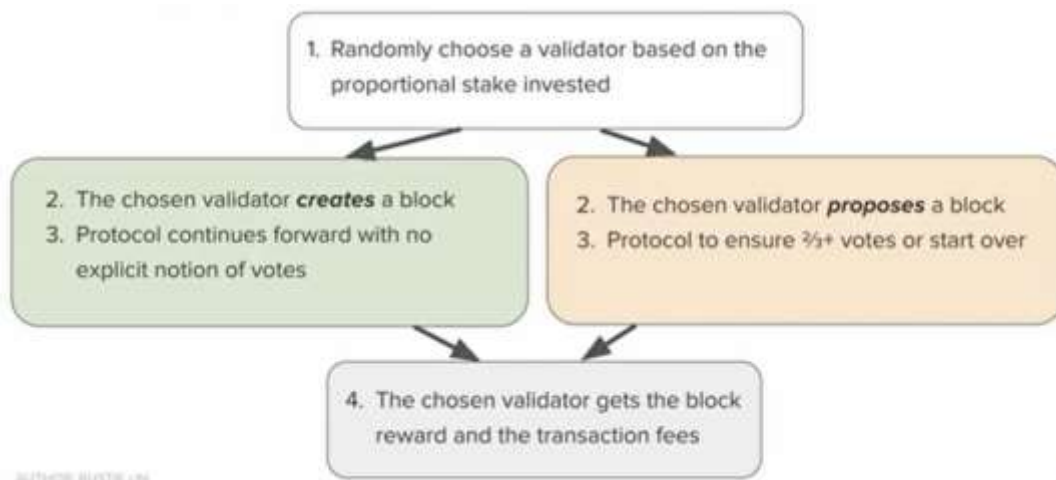
some sort of economic resource like in the case of proof of work that economic resources are your computational power and depending on how much in general of this resource (it doesn't necessarily have to be computational in nature) you commit is proportional to your chances of winning a lottery to make progress on consensus so in the case of bitcoin and proof of work you solve proof of work and you end up being the first miner to solve it then you essentially won that lottery and you get to put up the next block and others vote implicitly for your consensus output by including that next block in their version of the chain, that means that when they see you put up that block they're like okay checks out to me I'm going to stop my current proof of work and start a new one on top of this and we can really start to get creative with what resource am I requiring you to consume to participate in this lottery we could do CPU cycles but in bitcoin we've already got this where you can build CPUs that are optimized for solving these hashes so you know maybe that's not the best because that doesn't decentralize really well it goes to the people who can afford these ASICs so what else can we like have people commit and you can start playing around with some ideas like maybe currency why not just put money into it if you lose money that should stop you from you know acting dishonestly you could commit time as a resource maybe space or reputation which is an interesting one because that's sort of leaning towards centralization but the whole idea is as long as we spend resources to partake it should be economically infeasible for us to outspend the entirety of the network and that's the constraint that allows us essentially not to spawn up enough byzantine generals to overthrow the network. the most popular alternative consensus algorithm is something that's called *proof of stake* where instead of saying okay I'm gonna force my like

CPU to solve this hash a bunch of times instead you say you know whatever this native currency is (let's say bitcoin) I'm going to lock up some amount of my bitcoin when I'm a miner and then as I'm mining blocks if I make a mistake or act byzantine, if I try to change the contents of a block like readdress some transactions to myself or something like that or I use improper signatures, if I produce an invalid block I can get slashed which means I'll lose whatever amount of bitcoin that I staked and this way your probability of being chosen as a block proposer (as someone to put up the next block) is dependent; it's proportional to how much stake you put up so you have to commit a lot of this stake (you have to own a lot of bitcoin) if you want to effect consensus.

it's not very easy to just spin up a bunch of identities and take control of the network you need to own the currency and you're also liable to lose it if you start misbehaving similarly to how you would forgo bitcoin rewards if you were a bad miner. there's a couple of common implementations of proof of stake: in the first we randomly choose a validator instead of miners but depending on how much stake you put in let's say you put in 20 of the total stake you've got a 20 chance of being chosen as a validator and at that point you would be able to create the block we sort of take it as is there's no explicit voting and if someone were to disagree with you they would have to specifically not include your block in their chain and they would also have to essentially put up a proof on chains saying I can prove that put up an invalid block and as such their state should get slashed for it.

in the second version of proof of stake there's more of a deliberate like voting process encoded into it where again if you put in some amount of stake you might get chosen as a validator but at this point you would just propose a block and

then there would be another small subset of like voters who would also have to put stake in and they would take a look at the block that you propose (run the scripts on it, make sure that it's valid) and as long as there's at least a two-thirds passing vote from these voters then your block would get committed to the chain but so here there's sort of like an extra phase of this explicit voting process and at the end of the day if your block does get accepted by all the other nodes in the network and it starts getting built on top you'll get a block reward and you'll get transaction fees.



very similar to bitcoin in the payout mechanism but the punishment mechanism is a little bit different you'll lose your staked currency instead of wasting CPU cycles and what's nice about this is it's less energy intensive, it's not that I need to own a lot of computational power, I just need to own a lot of the native currency. **proof of authority:** rather than committing some amount of computer cycles like spending a lot of time hashing or spending native currency and if I mess up you can take it away from me you could just put up your reputation as stake this is called the proof of authority network. this consensus is like very similar to centralized systems because there's nothing securing it other than I am trusting this node with a given specific identity to properly do

my transactions to put together the block in the right way and if they don't do that you'd still be able to see if the node created an invalid block but there's no explicit economic punishment here much, as just their identity (their reputation) would be tarnished and they would not be chosen as a future node for verifying blocks. you can see this is less secure and this is why it will be much faster because you don't have to do proof of work where you're crunching all these hashes and you don't have to go through these voting rounds and things like that and proof of stake, it's just like running a server and people will trust your server. if you just want to get your chain up and running off the ground because it lets you run just like a couple of trusted servers if you are like the one running them and sort of building that into consensus it's very easy to end up with a centralized network here because you're literally saying here are the nodes with authority that I will consider to make valid blocks but it's good for testing and non-production uses.

```
// Clique is the proof-of-authority consensus engine proposed to support the
// Ethereum testnet following the Ropsten attacks.
type Clique struct {
    config *params.CliqueConfig // Consensus engine configuration parameters
    db      ethdb.Database // Database to store and retrieve snapshot checkpoints

    recent *lru.ARCCache // Snapshots for recent block to speed up reorgs
    signatures *lru.ARCCache // Signatures of recent blocks to speed up mining

    proposals map[common.Address]bool // Current list of proposals we are pushing

    signer common.Address // Ethereum address of the signing key
    signFn SignerFn // Signer function to authorize hashes with
    lock sync.RWMutex // Protects the signer fields

    // Authorize injects a private key into the consensus engine to mint new blocks
    // with.
    func (c *Clique) Authorize(signer common.Address, signFn SignerFn) {
        c.lock.Lock()
        defer c.lock.Unlock()

        c.signer = signer
        c.signFn = signFn
    }
}
```

Proof of space

there's some really interesting other sort of resources you can ask nodes to commit such as even digital like computer space or disk space; for example there are decentralized storage networks like IPFS the interplanetary file system and storage.io(you can think of them like a google drive but a decentralized google drive) so it's not just your files on some google database

somewhere it's your files are distributed across like a network of personal computers and to participate in consensus these computers have to stake that actual disk space that they have. Let's say I'm committing this disk space to consensus, users will actually be able to store their files in this disk space and if I'm a huge node (if I'm committing a lot of disk space to the network) then I'll have a larger role in consensus and that makes sense because the native currency or at least the base unit of value in a storage network is the actual amount of storage that it can support and this is also like a capped resource, you're not going to have infinite disk storage that you can throw at consensus in the same way that you might be able to make an infinite number of public private key pairs and like an infinite number of identities. for any consensus algorithm that might just give you sort of a one identity (one vote). this is still good under the constraints of Nakamoto consensus, it is economically and feasible to get all of the computer storage of this whole storage network.

Proof of elapsed time another interesting consensus mechanism one that could be a strong primitive if it gets implemented in a fully decentralized way and that's proof of elapsed time which is saying look you want to participate in consensus what's the one finite resource that we all have time if you can prove that you spent some amount of time that you had to lock up your operations in order to participate in consensus to put up that block or verify someone else's block. that's a pretty strong resource that you can commit for consensus and there are implementations of this where you just ask your miners to wait and just hang but what this requires is something called a *trusted execution environment* which is like essentially a specifically designed CPU made from a trusted manufacturer that in a verifiable way you know that

it will wait some actually random amount of time and you can prove how long it waited around. the issue here is that we're relying on this manufacturer to implement this hardware in the way they say they do but this is a really interesting primitive if we could make this work in a decentralized way like really prove that you spent some given amount of time during consensus this could be really strong. *Federated*

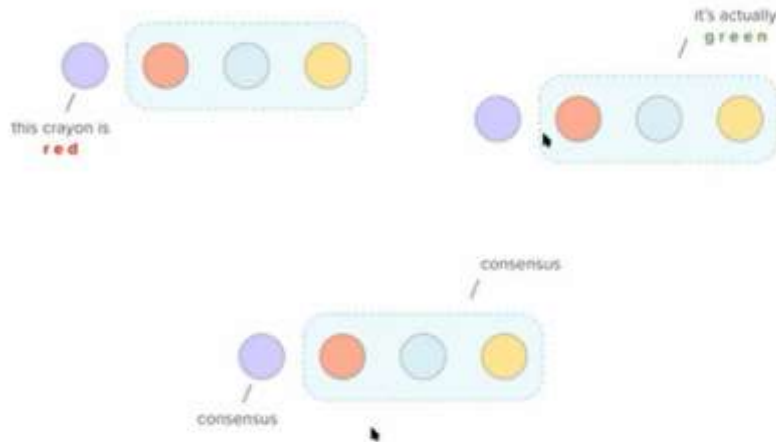
consensus

in a distributed system a quorum is a set of nodes sufficient to reach agreement like we saw within the pakistani parliament quorum is just a majority and once you have this majority that's really the first step towards coming to a consensus or agreement. in different networks there are hundreds and thousands like such large numbers of nodes that are distributed all over the world that in some instances it's virtually impossible to actually know all the nodes in a network so how can we still achieve consensus if we don't know a lot of people in a lot of nodes in the network? the idea of federated byzantine agreement is that we can slice up our quorum in general or our overall conform or group of consensus that we want to find so that we can convince a certain node or device or individual to agree with us and through that process then we can get everyone to get on the same page and come to consensus. Let's go through the little example here.



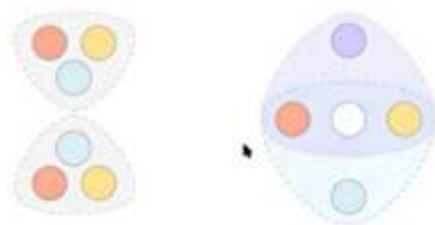
the red blue and yellow nodes are all in agreement and now this purple node comes along and asks this group of nodes who are all in agreement is this crayon red? so we're trying to come to agreement here that being said in response all these nodes

respond that this crayon is actually green and because of this purple node trusts these nodes here then the purple node joins on board and then comes to consensus.



we've now created consensus due to the fact that purple the node trusts this group of consensus here that we have and this will introduce the idea of quorum slices.

quorum slices are like small groups of consensus and we can actually come to see that quorum slices can overlap and rather than having to trust everyone involved in network we can instead just trust certain smaller groups.



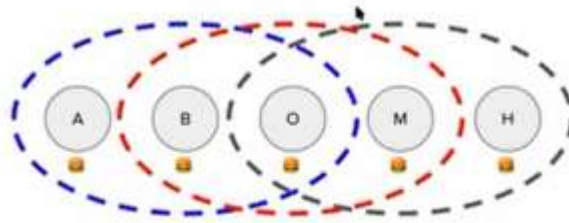
if we want to join a few different quorum slices together we can achieve quorum intersection and it looks like we have these two disjoint sets of quorum slices and in so the purple node trusts these three nodes (red and white and yellow nodes) and

this blue node also trusts these three (red and white and yellow nodes) so by trusting your friend essentially we can come to an agreement between all these different nodes due to the fact that if I were the blue node, I trust what red white and yellow say.

Let's have an example to clear the subject: going back to the initial lunchtime example here let's say it's lunchtime and we're about to grab lunch and we have Brian Oscar and Michelle so Brian wants to grab lunch and brings Oscar and Michelle along and Brian asks if pizza sounds good for today? Oscar and Michelle note that it's one dollar's burgers and it definitely makes sense to go check out those one dollar burgers now that Oscar and Michelle both want to get burgers then they're able to convince Brian to join on board with them and now we have a consensus and all these three nodes are all down to get burgers. now Hannah comes along and wants to join is wondering if she can also get some burgers but that being said Hannah doesn't really know Brian (they haven't really overlapped at all or met) so Hannah doesn't necessarily know if she can trust Brian that being said Hannah, Oscar and Michelle all form a quorum slice because they all know each other, they're all buddies, so it works out that they can all grab burgers together but if you note from earlier we'll see in a sec now Andrew comes along and he also wants to join a grab lunch but Andrew only trusts Brian and Oscar and because of that Andrew and Brian and Oscar all form their own quorum slice and we can see that these two overlapping quorum slices here with Oscar in the middle and if we remember we had this initial scenario of consensus between these three nodes as well, if we see here with these three overlapping nodes even though Andrew and Hannah joined a little late and may not necessarily know everyone involved we have these underlying

layers of trust that we can rely on to come to a full lunchtime consensus.

Lunchtime Consensus!



Lecture 7

Ethereum and smart contracts

<https://www.youtube.com/watch?v=GW7L1IAU9oA&list=PLLkiRvMHnp6OIWkZVa85g2tv7NtlgoAID&index=7>

Smart contrast

let's start off with smart contracts, but before that let's have a review about the question What makes Bitcoin special? what makes bitcoin so special is that bitcoin has four key components, it has one identity, two record keeping, three transaction and fourthly consensus, these four main components are what differed bitcoin from everything else at the time, but mostly it's a distributed network. Let's talk more about distributed network properties.

1. First it's pseudonymous if you remember, when we talk about bitcoin we emphasize the importance of identity especially in order to enable authentication and integrity so we have cryptographic identities that allow for accountability which gives everyone a unique identity through a hash and this generates a private key and also a public key.

2.two it's democratic decisions are made through a consensus protocol if you remember it's proof of work that is being used in bitcoin, so it doesn't require you to trust anyone on the network and you just need to trust the math behind the consensus.

3.thirdly it's immutable, everyone sees the same version of truth, since we have consensus and that means it's also temper evident data structure so that no one can change the history and what we do is that we use the merkle tree structure to compare the hashes in order to see it firstly.

4.it's unsensible to censor transactions, you need to own 51 of the network, which we talked about, is pretty hard and you wouldn't really want to do it because you won't profit a lot or at all, so it cannot be controlled by any one party

5.it's distributed, that means there's no central point of failure and the execution of transactions on the network depends on the network of miners located around the world.

- **Pseudonymous**, cryptographic identities allow for accountability
- **Democratic** decisions made through consensus protocol that **doesn't require trust**
- **Immutable** ledger of truth
- **Uncensorable**, cannot be controlled by any one party
- **Distributed**: no central point of failure

When we talk about Ethereum, keep um in mind of bitcoin's architecture. since we're talking about smart contracts let's just all agree on what traditional contracts mean, a traditional contract must be agreed upon and enforced by law.



thus we need a consensus protocol and through a consensus we should be able to agree on both the contents of the

contract and the execution of the contract, so what makes a smart contract, a smart contract is a piece of code that facilitates, verifies or enforces the negotiation or execution of a digital contract, in order to reach consensus trusted entities must run this code and we all need to trust that this digital contract is enforced correctly.

smart contract

(noun) /smärt 'käntrakt/

1. code that **facilitates, verifies, or enforces** the negotiation or execution of a digital contract.
 - a. **Trusted entity** must run this code

the difference from a traditional contract and a smart contract is that the execution and enforcement is done through algorithms and not through law.

Ethereum

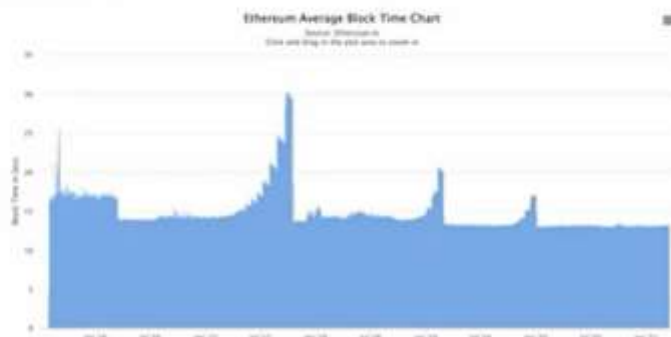
What is Ethereum?

what is the Ethereum on its website it says Ethereum is a decentralized platform designed to run smart contracts, it's a distributed computer spread amongst a multitude of nodes across the world that executes code that people feed into it and because of the specific architecture it can run many applications and the execution of transactions basically bring us from a previous state to a new state and that's called a state transition function.

Ether is Ethereum's native asset, it's basically the value in Ethereum ecosystem and it's not because of investment or anything but it's crucial in aligning incentives in Ethereum. bitcoin takes 10 minutes for a block to be created whereas Ethereum takes 13 seconds which is a huge difference and the exchange rate.

Misc. Implementation Details

- Block creation time: ~13 sec (Ethereum) vs ~10 min (Bitcoin)
- Exchange Rate:
 - \$367.75 USD (2020-10-16)
 - \$1788 USD (2021-03-18)



Bitcoin vs Ethereum like we mentioned a few slides before we know that bitcoin is special because it's the first successful cryptocurrency, it's trustless, it's immutable, it's unsensible, it's pseudonymous, there's no central point of failure, it's one CPU one vote or through proof of work but Ethereum has all these values but has a different use case. bitcoin you might have heard that it's the gold standard of blockchains and the asset is bitcoins. it's been around for the longest and has successfully supported many number of transactions, it's simple and robust it's UTXO- based (the piggy bank analogy) and it's also not touring complete meaning it can't have a loop, and the main purpose of bitcoin is to allow transactions of bitcoins. how about Ethereum?

Ethereum is a smart contract blockchain platform it's to execute these contracts that are written and it's seen as a distributed world computer and the asset is ether, it exists to fund computation and also align incentives and the main purpose is not just a global payment system like bitcoin but it supports a little bit more features like running programs on Ethereum for example decentralized apps (Dapps for short) and it uses a tiering complete which means it's able to write a loop

which means it's more developer friendly and there's more applications that can be written.

Bitcoin	Ethereum
<ul style="list-style-type: none">• The "Gold Standard" of blockchains• Asset: Bitcoins<ul style="list-style-type: none">◦ Primary purpose of the Bitcoin blockchain• Simple and robust• Stack-based, primitive scripting language, not Turing-complete• UTXO-based	<ul style="list-style-type: none">• Smart Contract Blockchain Platform• Asset: Ether<ol style="list-style-type: none">1. Fund computation2. Align incentives• Complex and feature-rich• Turing-complete scripting language• Account-based

let's look a little bit into Ethereum accounts. in bitcoin accounts private keys are there to prove ownership of a UTXO, when you spend bitcoin, you spend from the previous transactions and to calculate the current balance you have to sum up all your UTXO or all your piggy banks that you own and this makes it easy to make a transaction and prevents double spending but in Ethereum we don't use UTXOs we use private keys which proves ownership of accounts and tracks a current balance and this makes it more space efficient than UTXO as when calculating your own balance you just reference your own account which is stored next to the address in the state and you only need access to your Ethereum wallet.

it's much cheaper and it's easier to look up an account balance and transfers between accounts when we have an account model.



There's two types of Ethereum account models first it's the externally owned one and second is the contract account one. what's the externally owned account? it contains an address that they use to let people send them ether and also has a balance of ether that they own and it's owned by an external entity and sends transactions to transfer ether or trigger contract code. second there's contract accounts it's owned by smart contracts, it contains an address associated with the contract and persistent storage and the code is basically executed when externally owned contracts or other contract accounts make transactions to trigger the code's function calls. it's basically just triggering the whatever code it is for the contract account.



Ethereum smart contracts: control smart contracts are like autonomous agents that live inside of Ethereum network, it reacts to the external world when they are basically poked by the transaction which cause specific functions in the code so it has direct control over first internal ether balance, the internal contract state and thirdly the permanent storage. to wrap up this part what are the main purposes of Ethereum or smart contract? data represents something useful to users of other contracts for example it could be a smart contract defining a new token currency or an organization's membership which you have to pay a certain amount to obtain two. smart contracts can be used to manage a contract or relationship between untrusting users for example there's a lot of use cases of insurance companies, may be using smart contracts in the future for like rent or financial contracts, etc. thirdly it provides additional functionality to other contracts, it writes contracts that cause other contracts, perhaps using them as a software library or leveraging the functionality of an existing contract. fourth smart contracts can be used for complex authentication. **the recipe for mining: Ethereum** first you gotta download the entire Ethereum blockchain and then you have to verify incoming transactions and run smart contract code invoked by the transactions and then you have to create a block and then find a valid announce which is because we use proof of work still, eth2 however is moving to proof of stake which is probably more environmental friendly, and then you broadcast your block and then you profit. just to recap Ethereum is a distributed computer, it uses or currently uses the consensus protocol of proof of work and that's when miners competitively create blocks by executing code and searching for solution to the mining puzzle, the network

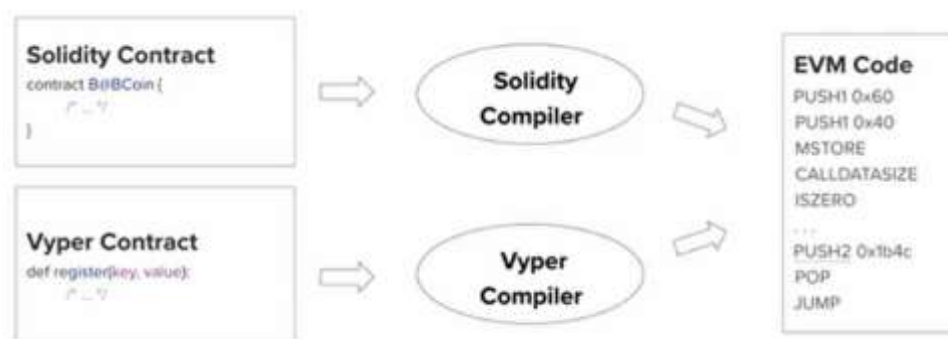
consensus removes the need for trusted third party and it's their secure peer-to-peer agreements that live on the blockchain forever.

What types of transaction model Ethereum uses

- A) It is an accounts based system - Y
- B) It is a UTXO based system
- C) Its very simple and robust
- D) Its complex and feature rich - Y
- E) It is a smart contract blockchain platform - Y

Ethereum virtual machine (EVM)

Ethereum smart contracts are generally written in high-level programming languages. we have solidity and viper which are the two most commonly used languages that write smart contracts on Ethereum, the difference is solidity is a mix between c plus plus and javascript whereas viper is more python-like, smart contracts are written in these language and basically compiled down to an EVM code or you can think about it as a machine language thus it translates from a human readable language through a compiler, to a simple language for a machine to understand and execute.



miners in ethereum create blocks by executing the EVM code and search for a solution to a mining puzzle and every Ethereum node has to run EVM.

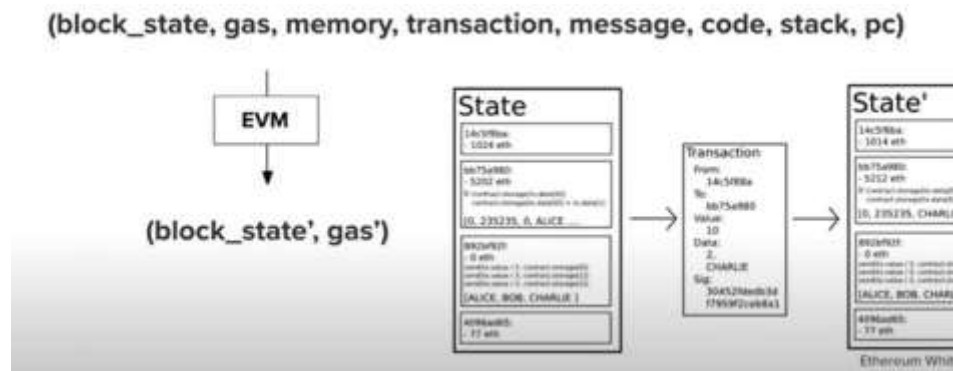
- The **EVM (Ethereum Virtual Machine)** is a "mini computer" on your computer that runs contract code
- Contract code that actually gets executed on every node is EVM code
 - **EVM code:** low-level, stack based bytecode language (i.e. JVM bytecode)
- Every Ethereum node runs EVM

what if our contract has an infinite loop? if there is an infinite loop, every node will be stuck executing the loop forever and by the halting problem if you learned in 70 or 170 it is impossible to determine ahead of time whether the contract will terminate and this leads to an easy way for attackers to launch a DOS attack which means denial of service and the denial of service attack is basically trapping computers around the world in infinite loop which means it's unable to execute other meaningful contracts. the Ethereum founders thought about this and that's why gas and Fees is a thing and it's the solution to this problem. gas is what fuels the execution of a given contract, it's what you have to pay the network in order for your transaction to go through so you can think about transaction fees in bitcoin equivalent to gas and Ethereum, so every EVM opcode requires gas in order to execute which prevents infinite loops and any denial of service attacks so every transaction specifies one start gas which is the maximum quantity of gas it is willing to consume and two the gas price which is a fee in ether the contract is willing to pay per unit gas and every transaction has to have this.

How Gas works

at the start of the transaction there's the start gas, the start gas times the gas price which basically represents the amount of ether paid for a computation so this is subtracted from the sender's account, the sender being the one who invokes the contract by sending this transaction, if the contract is successfully executed the remaining gas is refunded to the sender because not all the gas has been used up but if the contract execution runs out of gas before it finishes the execution reverts because you don't have enough gas so there's basically not enough power to keep going, but the ether paid is not going to be refunded because although the contract execution gets reverted someone on the network has put computational power to execute whatever EVM code you put and once the gas is spent up it proves that someone executed the program thus it's not refunded so there's two end states, one you run out of gas or two the program just terminates and even though a theorem still allows someone to write an infinite loop. it's possible to do it but the attackers attempting to DOS the network will have to pay enough ether to fund this and purchasing gas is a price you have to pay to use this distributed computational power, which costs a lot if you check up like gas fees right now it's going to be high. this disincentivizes users from running expensive computations without having sufficient funds so it will cost attackers a large amount of ether which is basically not worth it. **State Transition Functions** when we execute a transaction we go from a previous state to a new state. at the beginning there was just a blank genesis state which it's like the first block and then many transactions were executed after that and any point in time the final state represents the current state of the Ethereum blockchain. so at the start you have the current block state, the gas required,

current memory etc and basically everything you need to correctly execute a transaction and you feed this into the EVM and get out the new block state with all the updated account balances and states and new gas value and the gas that gets refunded too if it's properly executed.



What do STARTGAS and GASPRICE refer to?

- A) STARTGAS is a function in the contract called to start the usage of gas, and GASPRICE is the total gas spent
- B) STARTGAS is the maximum amount of gas the owner is willing to pay, and GASPRICE is the price of gas in wei per unit
- C) STARTGAS is a function in the contract called to start the usage of gas, and GASPRICE is the price of gas in wei per unit

B is the correct answer.

Conclusions: the main goal is not to optimize computational efficiency but enables distributed and trustless computation. every node on the network has to run the same computations which is basically redundantly parallel because they're doing all the same things on different computers and the contract executions are replicated across all nodes. the execution is very expensive, it is memory intensive and it is slow so it is not supposed to be efficient it's supposed to be a way for us to reach consensus among all parties and have a trustworthy output of the code.

- **Ethereum is not about optimising efficiency of computation**
- Its parallel processing is **redundantly parallel**
 - **way to reach consensus** on the system state without needing trusted third parties
- Contract executions are redundantly replicated across nodes
 - ⇒ expensive, slow, memory-intensive
 - creates an **incentive not to use the blockchain** for compute that can be done off chain

why would you use blockchain and when would you use a centralized database? you don't have to use blockchain for everything as much as the news tells you, use blockchain when you know you need for a shared database with multiple writers, mostly with parties you cannot trust and when there's no trusted third party or authority that is available you also use blockchain when you're interested in fault tolerance data immutability or censorship resistance.

you use decentralized database for when you're sharing information with parties that you trust, when you keep data confidential and we have to handle complex and large amounts of data because remember Ethereum is not efficient and you

need to be able to edit the data because again it's immutable so you can't really go back and change your name or any information and if you're interested in cost effectiveness speed or efficiency a centralized database is much better than a blockchain system so blockchain is not a solution to everything.

Use Blockchain:

- Need for a **shared database** with multiple writers
- Parties **cannot trust** one another, and no trusted third party or authority is available
- Interested in fault-tolerance, data immutability or censorship resistance

Use Centralised Database:

- Database does not need to be shared, or is shared by parties that trust one another
- Must keep data **confidential**
- Must handle **complex** and/or large amounts of data
- Need to be able to edit data
- Interested in cost-effectiveness, speed or **efficiency**

Use cases

State Machine this is a very common terminology that you'll hear all the time around blockchain world is that Ethereum is the global state machine, it's not really supposed to be a computer as much as it is a global state machine, there's a multitude of use cases that erupt from that, the first way to think about it is the global state machine Ethereum as the backbone for virtual currency so on Ethereum you can basically recreate bitcoin in four lines of code that's a pitch that obviously isn't entirely true, bitcoin has very intense optimized mechanics that make it very beautiful as an open source project but you can offer an ERC20 token on Ethereum very easily. there are boilerplate template contracts for you to create your own coin. if I wanted to make Ayush coin that a cool mechanism that every time I buy a shake from cream, I meant one more Ayush coin or something like that, I could do that on Ethereum pretty easily.

with one operation (with a couple lines of code) you can use the template, you can implement the ERC20 interface, just fill in a couple functions and you have your own token.

Obviously the token will not be useful to anyone if you don't think about the economics and think about the utility of the token but theoretically it is possible to do this in four lines of code. why do we create tokens?

if you create an application, not like a user application but a protocol application like a compound, you can use tokens to determine and affect the economics of the system that you've created.

- Token System Implementation
 - Recreating Bitcoin in 4 lines of code
- Database with one operation
 - Ensure Alice has enough \$\$ and that she initiated the transaction
 - Subtract X from Alice, give X to Bob

```
def send(to, value):  
    if self.storage[msg.sender] >= value:  
        self.storage[msg.sender] = self.storage[msg.sender] - value  
        self.storage[to] = self.storage[to] + value
```

Ether (ETH) is the native token used by the Ethereum blockchain and network as a payment system for verifying transactions. ERC-20 is the standard for creating smart contract-enabled fungible tokens to be used in the Ethereum ecosystem.

Public Registry the next use case thinking of the public state machine is public registry, there are many examples of this but one of them is a DNS system, there's basically a bunch of centralized servers and providers that do DNS lookup service for you. it'll map like a domain name to an IP address, this actually is very easy to implement on a blockchain because this is what Ethereum is made for, it's made to share a global state,

so it's more secure on Ethereum and it's cheaper because this might not be known but these centralized DNS providers actually do a lot of harm in terms of raking money from people who want to host their own IP or their own DNS or their own domain.

- DNS System
 - Maps domain name to IP address
 - "gillian.chu" → "12.34.56.78"
- Easy to implement in Ethereum

```
def register(name, value):  
    if !self.storage[name]:  
        self.storage[name] = value
```

there is a lot of kind of middleman corruption that putting this stuff on a blockchain would really do good. Ethereum might not be the best blockchain for this in practice because of the slow consensus time of proof of work and the expensive consensus time of proof of work. there are other projects like handshake protocol that have actually gone and implemented this.

continuing down this line of Who owns what, thinking as a public registry, Ethereum is very useful to tell you who owns what, it can be ownership of a private document, it can be ownership of some property that relates to an official stamp on the Ethereum blockchain, any kind of bureaucratic stuff that you can do in a normal bank or go to the city hall to see who owns what, you can do on a blockchain in theory.

"Proof-of-Existence"

- Proves ownership of a certain document without revealing it
- **Timestamps** verify ownership later

Use Cases (simplified):

- Rent server space to store documents
 - Proof document is unmodified via hash values
 - **Integrity** guaranteed

the world still has to open up to these ideas but in the next decade or two, where you'll see the biggest impact of blockchain is these kind of basic ownership ledgers, an example of this that i like to point to is land titles, for example right now all this stuff is done on paper in basically outdated ways. there's flawed paperwork unclear documentation, there is either a lack of central government authority, like rent control I mean my apartment here in berkeley is getting a rent spike of 15 and there's really no governing body to protect me from that when there should be but in terms of who actually owns the land title if I wanted to find that out I could go to city hall and I could technically find out my landlord's name and see if he owns it or not but I'll never do that because that is so tough.

because of that lack of transparency and all the steps that it takes to verify these things, I personally have no idea who owns this building because I will never go to check and most people don't. that opacity creates a lot of problems like bribes tampering with records and the government basically cannot support this land record authority.

Problem:

- Flawed paperwork, forged signatures, **unclear documents**
- Lacking **central** government authority

Pitfalls:

- Corrupt officials accepting **bribes**, **tampering** with records
- Government cannot support land record authority
- Citizens **mistrustful** of NGO/multiple NGOs



if you put it on a blockchain there are ways to tie the technological stamp of your land to an actual physical identification of your property and this is nothing new, you have an ID code for your land title, you have an address, you have already things that uniquely identify your property or your

assets or anything like that, these are perfect examples for blockchain because you have a one-to-one mapping of a unique identifier.

The Blockchain Solution:

- Hashes & Digital Signatures
- Transparency
- Immutability
- Limits Centralization



Simple mechanism to transfer ownership, like making a transaction on Bitcoin

why trust a bureaucratic and slow opaque system to do that when you can simply put it on a blockchain so this is where we see the next real use cases for blockchain popping up especially once we have institutional adoption, once people open up to the idea of crypto economies, maybe we'll start to see this in the future, personal blockchains and crypto will become a really important part of personal finance in the future there are countries investigating this Georgia, Ukraine, Sweden always eastern European countries are ahead of the curve in relation to us so one caveat here is that blockchain is only as good as the information fed into it just like ML, just like any data heavy technology, garbage in equals garbage out so you have to trust the entry points for blockchains, you can't assume that because it's on the chain it's trustworthy you have to understand where the information is coming from in the first place. that is really the main problem as to why this is not an easy fix, this is not an easy transition from our current system to a blockchain, it's because we have to create these data pipelines and that simply will just take time, if it was easy it would have been done already.

Microgrids shifting a little bit we can go towards use cases like microgrids, these use cases are very interesting because they are at the infrastructure layer where you would not ever have to interact with a blockchain here but the use of a blockchain can help you as the energy user. let's go ahead assume that we live 10 years in the future and we have the infrastructure to be able to redistribute energy just like internet packets and if we had a blockchain that is the only way we could achieve real-time consensus on who has what flow of energy in coming to their house, who has an excess flow, who has enough. if we were to rely on a centralized place for this that would be an almost immediate attack for like a cyber attack or a grid meltdown attack because that would be as easy as targeting one server like one like EC2 or something and that is not secure. eventually when we start to create this kind of custom infrastructure and create move towards smart cities blockchains will be a very big part of the way that we achieve very important security guarantees about our actual underlying infrastructure. this is another kind of category of use case, not necessarily solving coordination failure or making things transparent in public but actually provable security bounds for very important things like our energy grid.

Defi

the biggest and most useful blockchain use case in the next decade will be DeFi.

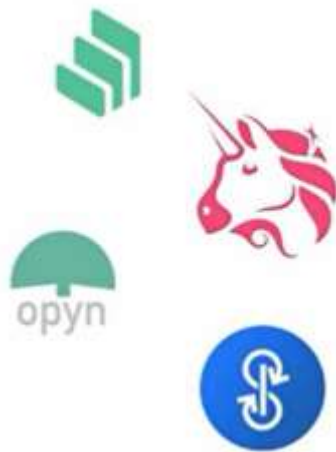
DeFi stands for *decentralized finance* and it is essentially a thriving ecosystem of financial applications that execute in a

decentralized manner so any financial primitive that you do with your bank in your normal stuff like lending for a loan, trading on forex, having insurance, borrowing and any financial primitive that you can do normally now you can do it basically on a blockchain and why do you want to do that? well for one it's much faster, two it's a lot more liquid because it technically is less liquid but the markets themselves move a lot faster and there are a lot more accurate predictors of what's going on in the market. the markets are a lot more transparent, most importantly it's decentralized so you won't be shut down if you're trying to buy game stock robin hood which is a centralized trading platform who is buddy with citadel won't shut down your trading because the hedge funds are losing in DeFi it's all based on protocols so the code will not magically make a decision to help one person or the other it just runs in an automated fashion because there is no central person making any decisions.

the big problem is that traditional finance is close to regular users and all these crazy derivatives and trading and all this stuff you can do are on centralized platforms that as we've seen in the modern day get censored so these platforms are not built for the average person to win, they are built for hedge funds and built for institutions to win and that's something that the public has more and more been aware of during these crazy times that we've just been looking at it more so they can manipulate markets which they do.

retail users are literally priced out and it gives power to the big banks so on Ethereum which is the core backbone for this entire ecosystem of decentralized finance you can build out protocol applications using smart contracts that handle this financial logic and the best part is you can go on Etherscan and actually see the very code that you are using to handle your

own money so it's a completely different paradigm than what we've ever had before you can actually verify and debate about the protocol and the game theory and economics that are used to handle your money so you have as much of a stake in these protocols as you want.



on the picture above there's a couple logos there but on the top is compound which is a lending company built on Ethereum and that is the most stable and very well built out lending company on Ethereum on the right there's Uniswap which is a very famous exchange you can swap the tokens on Ethereum opyn is an insurance platform started by a blockchain at berkeley members they recently raised three billion dollars for their seed round and on the bottom there is yearn which is basically an aggregator of all these functions.

Drawbacks

it is capital inefficient there are high fees right now because on Ethereum proof of work is expensive so to trade you actually have to pay gas fees and the miners, the users pay gas fees and it's more expensive than it should be hopefully this will be solved in the next couple year honestly because eth2 is going to come out ideally later to scaling solutions will come out which will make these gas fees reduce significantly but until that

happens it is inefficient. there also is risk from composability in smart contracts and for example if I wanted to build an application on top of compound I could do that I could literally call the compound smart contract and build on top of it and that's very encouraged in this space but if compound goes down so does my application. the benefit to this is that there are winners here so places like compound and Uniswap they are actually established enough and trusted enough that you could build on them without any worries but there is a risk you can't just blindly trust, you have to do your own analysis and then front running is a major problem in the industry right now which to be fair also happens in traditional markets as well like entire companies like jane street are basically built on the idea that they can front run with high frequency trading so this isn't anything super new but it still is a problem in this industry too. learning curve is high, it's getting better because more education on these things and the user experience has improved like a lot so you can actually use wallets now like metamask makes this all really easy with just a couple button clicks you can do it but ultimately you still have to have some idea of what's going on underneath and it's going to take a while for the majority of the population to be able to speak that language and that is kind of a fact of we're kind of on the cutting edge of things but for the entire 8 10 billion population of the globe to be able to be comfortable with blockchain it'll take time. **Prediction Markets** these have already kind of been put to use recently in the Trump Biden election, 2020 election, there were actually bets placed on who would win the election and the idea here is that this bet would be more accurate than polls, it turned out to be relatively true, arguably there are arguments on both sides. prediction markets are cool because you have stake in the bet so the idea is that because you're

putting money down on what you think will happen it's a lot more accurate indicator of the public opinion than simply saying yes or no on a poll from someone who you don't even want calling you in the first place. we found that in the election it did obviously show that Biden would win but the polls said something like Biden was a landslide victory candidate and the prediction market said that it was much closer than that so the disparity is interesting to research and look at. no one has really concluded either way because you can't, it's kind of hard to prove one way or the other but definitely interesting to look at how they're so different. oracles report the outcome so an oracle is basically a trusted pipeline to the outside world for a blockchain as I said before garbage in is garbage out so an oracle is like something that the community trusts to put good data in and it could be centralized, a lot of the times it has to be but there are companies that make decentralized oracles like augur.

Draws on the wisdom of the crowd

Ex: *Who will win the 2020 Presidential Election? Trump or Biden?*

1. Bets are replaced with shares of outcomes
2. Oracles report the outcome



Benefits of Prediction

"A service that never crashes, a service that's completely transparent..."

Benefits:

- No restrictions on market types
- Shared liquidity
- Censorship-resistant
- Automated and trustless

in a way you're buying information like what movie x flop, will my house burn down, is there a bug in my smart contract, so this is where you start to get really sketchy with prediction markets, since it's like decentralized and a company would never pose a question like will Ayush's house burn down like that would just like no central company would ever pose that as a question and have people bet on it but in a crypto anarchist world there's no rules of what you can bet on so this is kind of just fair game and this can even be taken to the point of really nefarious things like assassinations and things like that. it is just an interesting mindset shift to think of these things. So I bet one million dollars that Bob will be dead october 4th or be alive on october 4th, this could create an assassination market and that's kind of scary to some people so this is another reason why crypto and blockchain are scary to many people is because of these kind of fundamental shifts that take us away from what we consider to be a quote-unquote secure estate so this is the philosophical part of blockchain, it's not just tech that you use to make your product better it literally is representative of a fundamental shift in the way we think about society so it's not just for industry.

Lecture 8: Scaling blockchain

<https://www.youtube.com/watch?v=6TBNPVkATEc&list=PLLkiRvMHnp6OIWkZVa85g2tv7NtlgoAID&index=8>

Background

what is the scalability problem? we're talking about scalability in this lecture which is honestly one of the most interesting nuanced and modern parts of blockchain development right now and when you think about scalability and what blockchains are supposed to be in general you can see that blockchains offer all these sorts of services like decentralized execution environments and decentralized cryptocurrency but if the world and people across the world and everyone in the world can't use it how useful is it? the scalability problem is making sure that a blockchain is able to offer its services to everyone in the world regardless of how many people are there interacting with the blockchain itself.

you can see it very directly through things like if you've tried to make an ethereum transaction any time in the last six months where the common trope is that gas prices are way too high it's a very practical problem right now it's way too expensive to work with all these blockchains right now and scalability is how

we solve that. when we say providing all the services that a blockchain offers we can take bitcoin as an example and we know from the past lectures that bitcoin has privacy guarantees, it guarantees that you can be a pseudonymous user and what we want to ensure that as the network grows as more users join the network and as more people start to adopt bitcoin that all of these services are still guaranteed for all years of users so you want to make sure that all these services and guarantees are still in place independent of how many users there are. the common metric that people use when they think about scalability is this thing called TPS or *transactions per second* and the common metaphor and example given is let's say you want to go and buy a cup of coffee and if you're working with something like bitcoin if more than three people are trying to get a cup of coffee at any time it means that you can kind of see what the average TPS is looking, the bitcoin blockchain is very slow.

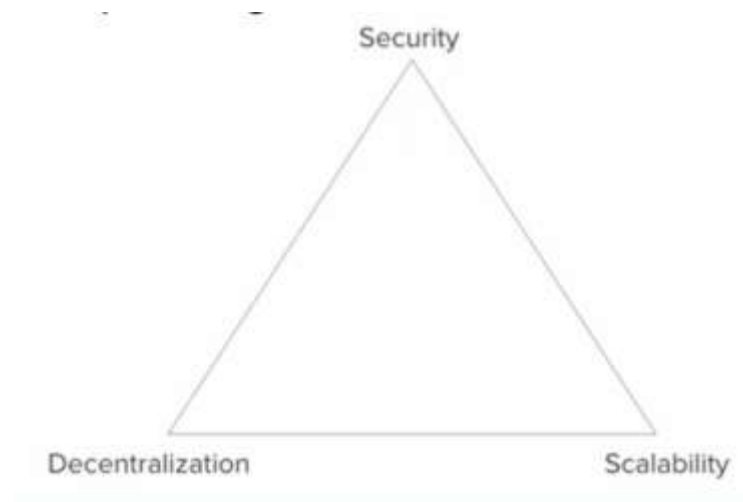
	Average	High Load / Maximum
Bitcoin	3 tps	3.2 tps
PayPal^{*,**}	150 tps	450 tps
VISA^{***}	2,000 tps	56,000 tps

people often like to compare TPS to things like paypal and visa it's unfair in certain ways but we'll take it and that's what we're aiming for when you go to something like TPS and right now this should be more than enough to visualize that we need to do a lot more when it comes to scalability for something like the bitcoin blockchain. now we're going to talk about all sorts of scalability solutions that are across different parts of the blockchain so we're going to discuss these two different sides there's a layer one and the layer two.

layer one is what you've been learning this entire time about the core blockchain architecture, the bitcoin blockchain and the ethereum blockchain that uses consensus mechanisms like Satoshi consensus plus proof of work and these are the exact blockchains themselves so changes made here to improve scalability on the layer one are made on chain itself we're changing the way that the blockchain works fundamentally then layer two builds an entire level on top of your primary blockchain so these create other networks for frameworks or protocols on top of the existing blockchain. these kind of build on top of the core blockchain which they rely on for security itself so we'll hear about a couple of key features and key projects that are layer two scaling solutions, so a lot of these changes are not directly on the blockchain itself but are other things that you can interact with that are still part of this entire blockchain ecosystem.

One thing to keep in mind is that the differences between L1 solutions and L2 solutions we'll be going into solutions that are in both categories and you'll see that with L1 solutions it's much more restricted in what you can do so when you're modifying the actual blockchain, you kind of have to stay within the realm of what that blockchain was meant to do, you don't have a lot of wiggle room when compared to L2 solutions when literally your imagination is the limit and whatever you can think of as long as it carries over those same guarantees from the base layer you can make it and if it makes the system faster while also keeping up security then that's great. that's why L2 exists in the first place because L1 changes are so restricted (it's so hard to make changes on these blockchains). another piece of background information is the scalability triangle. for scalability we like to divide it up in between three topics, first you need to ensure security for your blockchain next you want

to make sure that your blockchain is sufficiently decentralized and also of course in the scalability triangle you need your blockchain to be scalable.



we're going to go through a bunch of scalability solutions and show how each solution makes compromises on each of these variables on the triangle and kind of the way you can set up your mental model of how to use this triangle is any solution you can imagine as somewhere within the area of this triangle and some solutions might be more scalable but they might decrease decentralization, so for an example of this would be like the hash puzzle, the hash puzzle that miners have to solve when they're propagating a block and they're adding a block to the end of the blockchain, this hash puzzle currently is set to be around 10 minutes so for every block that is added it takes around 10 minutes to add that block and you might be wondering well if you want more transactions per second why not just decrease the difficulty of the hash puzzle if we decreased it from 10 minutes to one minute for example we already have like a 10 times increase in scalability but the problem with that is it decreases the security of the system and the main way of preventing the 51 attack which you may have heard is that it just takes so much work and so much costs to actually perform a 51 attack because the hash puzzle takes so

much energy and funds to actually solve, especially if you're trying to beat out the rest of the entire network, if you were to decrease the complexity of the hash puzzle then it would make the 51 attack almost trivial to just execute and that's very detrimental to the security of the system. that's just basic example of how one can increase scalability by a lot but also decrease security. the solutions that we analyze and evaluate are going to aim to increase scalability while also keeping up security guarantees and decentralization guarantees. *Layer one:*

the first layer one scaling solution that we will be talking about in this lecture is taproot.

tapper is a bitcoin improvement protocol and what that means is that this is kind of a proposal that a bunch of bitcoin core developers propose to the bitcoin core team and this is something they're working on currently this is just an example of how anyone can get involved with these serious significant changes that are happening at the bitcoin network. there are some very clear focal points that we can focus on when looking at what we can improve in the system to scale and one of the first things with bitcoin at least is signature verification, signature verification literally happens everywhere so many signatures are verified every second, the first case that we can see signature verification or a lot of signature verification is when new blocks are validated. when a miner is told about a bunch of transactions and when they want to put a bunch of these new transactions that are floating around the network into a block the miner has to go through each transaction and verify the signature of each and every transaction before putting into the block (from earlier lectures a signature is how you prove that your identity is the correct identity that is

allowed to spend some transaction of the network) and so the naive method of proving your identity is showing everyone on the network your private key, if you show everyone on the network your private key then everyone can see that this transaction was sent by Alice to this private key, I own this private key therefore I am able to spend this transaction; the problem with that is if everyone on the network knows about your private key then everyone on the network is able to access your funds and spend your funds and you would get robbed of all your bitcoin so we don't want that, instead of that we have signatures which is a cryptographic magic that smart people came up with so you don't have to show your private key but you're still able to mathematically or cryptographically prove that you are indeed the person corresponding to the bitcoin public key that the transaction was sent to. the first case where we see a lot of standard verification is when miners are validating new blocks and as you know root blocks are constantly being added at all times in a bitcoin network the second case is when a new node connects to the network so say I just found about bitcoin and I really want to get involved and participate in a network, I would go to the github and download the bitcoin core wallet and I would run through the installation process and during the installation process there's these couple of steps there's a step called IBD or *initial block download* and what happens in initial block download is as a new node I download every single block since the genesis block of the bitcoin network because I need to have access to it as this is under the qualification I'm a full node (full nodes download the full network) and when I download all these blocks from the past I'm also simultaneously verifying these signatures because as a new node I want to make sure that I'm you know downloading

the blockchain that is honest and true, to do that I verify the signature at every point and make sure that I'm downloading the correct blockchain and as you can tell signature verification is happening all the time but it's very slow and prior to taproot and taproot kind of fixes this. taproot fixes some privacy (you must remember in previous lectures about multi-signature wallets and may about bitcoin scripts, bitcoin scripts are expressions that can be evaluated to true or false and based on whether it's validated true or false it allows the user to spend a transaction or not and it's basically like a bitcoin's version of smart contracts it's not as robust as ethereum smart contracts but still allows for some logic to be involved) to taproot what happened was that in transactions are publicly available to everyone in the network and a transaction that included a bunch of bitcoin scripts and a bunch of multi-signature wallets and these were able to be observed and analyzed by anyone in the network (anyone can see which signatures were involved with the multisig wallet or which scripts worked on the condition of different signatures) and so all of this information was completely publicly available to everyone and as you can tell that leads to some privacy vulnerabilities (you don't want an attacker to know under what conditions you wanted a node to spend a couple of bitcoin or node wanting me to spend a couple of big planes) as a network we want to make sure that all of this information is obfuscated and is as secretly viewed as possible. what taproot does is it integrated schnorr signatures into bitcoin and store signature. it's a type of signature it replaced ECDSA signatures these are the signatures that are used in the bitcoin network to verify transactions and the reason that we incorporated signatures is because mathematically and cryptographically short signatures are better than ECDSA signatures in literally

every way. the reason that we didn't include schnorr signatures previously and the reason that is now being added is that anyone who developed signatures had patented it and so we couldn't until recently that the patent was lifted or something and then it allowed bitcoin core developers to add it. taproot also integrated merkelized abstract syntax trees or MAST (you might be familiar with the term merkel who learned about merkel trees back in earlier lectures) back then merkle trees were used transactions and making sure that the merkle root was unchanged to make sure that the blockchain was unchanged but now we're using them a little bit differently but we're still using that same underlying data structure.

one of the main ideas of taproot is making signature validation faster because if we're able to make signature validation faster then not only do we decrease the barrier of entry for new nodes because we don't have to spend as much time verifying billions of signatures it also makes everyday use of the bitcoin blockchain faster and it increases throughput because miners are validating signatures by second basis and they're able to do it faster so this kind of propagates throughout the rest of the network.

the way that it's able to do this is through short signatures, the schnorr signatures and through some cryptographic, short signatures offer key aggregation and also verification in linear time what this means is the benefits that short signature brings specifically are: it allows for multi signatures to look the same and it costs the same as a single signature, this is pretty huge because more and more multi-signature wallets and most multi-signature transactions are becoming more and more popular an example is if you have a coinbase account and if you have a coinbase wallet well that's a multi-signature wallet

it's like a two of three multisig wallet where you own two of the keys and coinbase owns one of the keys and this is like a fault tolerance if you were ever to lose one of your keys then coinbase could step in (if you were to lose one of your keys and it was just you by yourself and you needed two keys to unlock the wallet because it's a two of three wallet then you would be screwed because you lost one of your keys you can't unlock the wallet because it requires two but you only have one so then all your funds and that one will be gone but what multisig allows is coinbase has one of the keys so if you ever in a scenario that you lose one of your keys coinbase can step in and help you unlock it) as you can tell there's so much more functionality with these wallets and they're becoming much more popular but the idea behind tapper is that it makes verifying these multisig transactions, costs the same as a single transaction which greatly increases efficiency and if any attacker is trying to maybe map out the network and see which multisig wallets are going where and which addresses are linked to other addresses, they want to be able to do so after chapter because mostly what multi-state transactions look the same as a single signature or a single transaction what store also brings is batch validation and previously signatures had to be validated in a serial manner, you had to do one after the other but now this allows for parallelization of signature validation and if you're a computer science you know that parallelization makes everything so much faster. with the onset of batch validation you can tell that um not only are new nodes able to join the network quicker but also miners are able to propagate new blocks faster which increases the efficiency of the entire network and also an interesting note about the bitcoin scripts is that in one single transaction you could include many bitcoin scripts say I'm giving niche some money

but I wanted to only use the money after two weeks or only use the money after he has zero bitcoin left so whichever one of those comes first then he'll be able to access the funds so before taproot anyone on the network would be able to look at both of those spending conditions (both of those bitcoin scripts) and be like okay that's pretty interesting you know maybe niche is poor and he's losing some money and that's why I am trying to give him some money when he reaches zero or there's some other assumptions that can be made after taproot now only the spending conditions that's actually activated and results in true so only the spending condition or the bitcoin script that allows niche to access the funds is actually publicly shown on the network as you can tell this leads to some privacy benefits. all these different transaction types multisig transaction, transaction with scripts they just look the same as a regular so for anyone who's trying to map out the network (because that's an actual thing that attackers do they want to see which transactions are going where which addresses are sending to other addresses more frequently because all these are potential attack vectors) taproot makes everything look the same and attackers are very confused and they don't know it's just reducing the amount of knowledge information that attackers have which is by far one of the best ways to prevent attacks and all scripts of a transaction are not broadcast to everyone you see in terms of functionality because evaluating signatures is so much faster we can now have much larger cap and multisig transactions. you might be wondering why this is useful for example with the dial you have people all over the world globally interacting with a wallet and you have condition where if two-thirds majority or super majority agrees on something then you can access funds to actually use those funds beforehand that would take a lot of

compute power and a lot of time and energy to accomplish but now with the benefits of tap root that's much faster. it's a huge scalability plus and we can now use much larger scripts because not all scripts are broadcast on the network only that one script that's actually evaluated are there.

Segregated witness

Segregated Witness (SegWit) is a Bitcoin transaction change that separates transaction signatures and scripts (witness data) from inputs and outputs data

Let's talk about What problems did segwit solve, well before segwit there was this issue called transaction malleability (it means that you can imagine a transaction like the essential parts of a transaction are the inputs (how much bitcoin is being inputted) and the outputs (how much output bitcoin is being outputted) and who's sending it to who; so those are the essential parts of the transaction that really define what a transaction is) the thing with transaction malleability is inside of these if you look at transaction raw data you can see that there's a field that wasn't essential to the transaction but it could still be modified (a couple of characters could be changed) and why does this matter? we use a transaction as an input to a hash function and as you learn with hash functions they have this thing called the avalanche effect where if you change even one bit in the input then the output is completely different so you can imagine what we want on the network is for two transactions that do the same thing (that send the same amount of money from Alice to Bob) to have the same ID but what transaction malleability shows is that when people modify this single bit that wasn't essential to the transaction then the transaction ID or the hash would be completely different and so what you could have is like multiple copies of the transaction in the bitcoin network that all did the same

thing (sent the same amount of money from Alice to Bob) but had different IDs and this leads to a network flood attack where an attacker could just spam the network with a bunch of same transaction essentially just like slow the network in that way so that's a very roundabout definition of transaction malleability and that was the main problem that would fix in addition to transaction malleability segwit also fixes a very inefficient version system, the script versioning system, refers to how bitcoin scripts were updated; bitcoin scripts is like a programming language and program languages change over time and the same thing has those bitcoin scripts but the updating process was very inefficient and very hard to actually use in addition to that there were also some inefficiencies regarding signature hashing operations and also we saw signature data everywhere. the reason that we don't need to see signature data everywhere is because when we see that a block is a hundred blocks deep from the end from the very tip we assume that it's valid and it's been verified we trust that the miner and other people in the network have successfully looked at the transaction inside that block and verified it, so we don't need to keep the historic signature data inside those blocks rather than keeping the data we could include more transactions.

And there is the raw data of a transaction here:

Before

```
[ ... ]
"vin" : [
  "txid": "8627052b6f28912f270306a912ea577f2ce4da6caa5a5fbd1",
  "vout": 0,
  "scriptSig": "<Sehyun's scriptSig>",
]
[ ... ]
```

After

```
[ ... ]
"vin" : [
  "txid": "8627052b6f28912f270306a912ea577f2ce4da6caa5a5fbd1",
  "vout": 0,
  "scriptSig": "",
]
[ ... ]
"witness": "<Sehyun's witness data>"
[ ... ]
```


this is a simplification of what a transaction actually looks like on the bitcoin blockchain. we have a vn field and that's the inputs with the transaction ID and we have this vout and inside this vout field we have a script sig field, so what this script field meant is that this is where we hold our witness (this is where we put our signature to prove that we are able to use this). the field (<Sechyun's scriptsig>) is not essential to the transaction if you think about before the aspects of the transactions that are essential to it are the input, the output and who's sending the money to who and this doesn't have anything to do with it and as you can see if you were to modify even one character in the script sig then the entire transaction ID would change because we're passing this entire bit string into a hash function and if even one character in here is different then it would result in a community of ID but now afterwards we've moved this script sig outside of this vout field and now that field is pretty much outside the transaction or outside of the essential parts of the transactions that's actually hashed to create the identity of the transaction. what happens when we move it outside is even if we were to change anything here then we're not modifying these fields inside the actual transaction and so we know that the essential parts of the transaction are what's hashed and that won't change the transaction and ID will stay the same. we've kind of solved the tracking the transaction malleability problem and this is the main thing that cypher did. it moved the witness data outside the transaction.

You might be wondering if you move a signature outside the transaction then is it not found in the blockchain at all? well it's still there just in a different structure and this is where we'll see this merkle tree structure pop up again, in the original merkle

tree that we learned back when we were learning about what a block contains we learned that merkle trees and the nodes contain the hashes of transactions and those hashes are kind of compatible together to make the upper layer and these hashes are propagated all the way up to the merkle tree. we can replace the transactions with signatures and also we can mirror the structure of the transaction merkel tree so the signatures match one to one so that you have: A signature transaction B transaction and B's feature candidates all the way up and you're able to formally prove that a signature of a corresponding transaction is actually included in merkle tree through a merkle proof and that's how we can keep the validation properties part of the blockchain. with script version and script and signature hashing these were more minor changes but also very important and the way that bitcoin scripts were updated previously was that the op codes that make up a bitcoin script there's about 10 to 20 op codes and the way you would update bitcoin scripts before segwit is you would have to take an opcode delete it and then overwrite it with a new op code that would perform the functionality that you wanted as you can tell this is a bad way of updating scripts maybe you want to keep all the options and just add a new one or maybe you never know down the line when you'll need an op code so this is not the most ideal way of implementing updates what segwit actually allows now is an introduced version numbers so that upgrading to a new script version could be done simply by increasing this version number there's some details abstracted away but basically we just simplified updating scripts and segwit is implemented in the bitcoin network and it's completely integrated. it was integrated much before taproot and taproot takes advantage of this update to script versioning.

when bitcoin core developers said they wanted to add short signatures to bitcoin they could just use this version number or this conversion system and just change the number to a version that they wanted to incorporate short signatures into transactions. we already see the benefits of segwit happening today in terms of scalability and for signature hashing this is much more typical stuff but what happens is that it resolved the issue of quadratic time signature hashing so when you double the number of transactions or you double the number of signatures but that would result in quadrupling the cost of validating the signatures because data was quite good but now it only verifies each

byte(it only hashes each byte at most twice) so instead of quadratic time it's two end time.

Now we go into the pros and cons of segwit. the pros is that it fixes transaction malleability (remember we move the witness data outside of the transaction and so transactionality is fixed and the reason this is so huge is because this actually paves the path for a lightning network to come into play and lightning network is an L2 solution that we'll talk about later on and you'll see why this is so used because lightning network can really change the face of bitcoin additionally) segwit was a soft fork and it didn't require the entire network to fork which is a hard for it. bitcoin corridors really do not like hard forks because it divides the network and you want the network to be as unified and empty as possible for decentralization purposes. it increased efficiency as you can tell through the sig hash operations and also the script versioning. it did increase block size, part of the segwit update was to increase the block size by a couple of kilobytes or maybe like one or two megabytes, not too much but very minor blocks has increased but still you can fit more transactions in there so that's a scalability increase

and also result in a smaller size of blockchain because you don't have that historic signature data found everywhere. in terms of cons it was just a one-time linear capacity increase for block sizes. it's cool for a little bit but we'll need more stability solutions in the future.

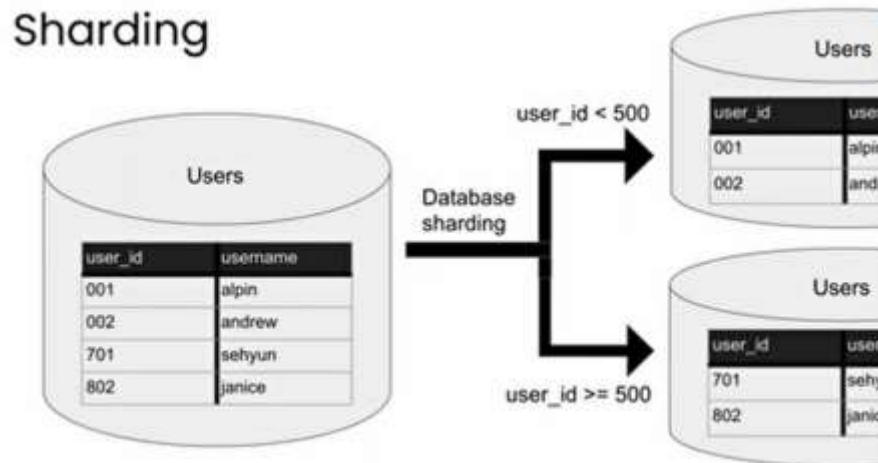
there's also obligated all wallets upgrade so all nodes had to take the responsibility to update their software or their wallet software to be compatible with segwit. even though segwit was backwards compatible with unspent transactions that were pre-segment (nodes are still identified) because it has a lot of all these benefits also better ways to solve transaction malleability and this isn't the most optimal or ideal solution by any means but bitcoin core developers have to keep into account the wants and needs of all three users or all three parties of the bitcoin blockchain so developers and miners and regular users have different needs and they have different pain points and so this was a compromise or the best compromise they could come up with to solve transaction malleability.

Sharding

now we're going to talk about another L1 blockchain improvement which is sharding this is getting into completely revolutionizing the way that a blockchain is designed.

sharding should kind of harken back to other mechanisms in computer science such as database sharding so we have a little diagram. if you have a huge database and you're trying to make queries (the meaning of a query in database management is a request for data. If you need to access, manipulate, delete, or retrieve data from your relational database, you'll need a database query written using a specific syntax.) to it if the entire database is here and you have to manage every single

query for every single piece of data in the entire database it can get a little overloaded but the improvement that you can make for things like databases is sharding a database. You can split up so that half of the user IDs you go to one specific subset of the database which has its own query response system or if it's greater than 500 the other half you go to the other shard of the database and it's very similar for blockchains as well.



in blockchain sharding is the idea of not requiring every miner to be working on every single block, essentially creating parallel but connected blockchains. every single block in a normal blockchain is to be managed and every miner is responsible for every single block in a traditional blockchain but in a sharded blockchain a blockchain is instead split into a bunch of sub-blockchains that together will make up the entire blockchain state, so each sub-blockchain have its own set of full-knowns minors that will manage the consensus and block production and propagation for that singular blockchain so in that case you're almost creating this multiplicative effect of having like a 10, 20, 30 shards, basically the number of shards you have times improvement in the scalability of your blockchain.

there are a lot of things to keep track of when it comes to sharding you're gonna have a bunch of different types of nodes, you might have nodes that are simply responsible for one shard, you might have systems where nodes will hop out and nodes validators will hop around different shards like near, you might have other nodes that manage different shards themselves or super full nodes which manage the entire blockchain, when it comes to sharding a couple of challenges appear that we'll list a few of them:

the biggest one when it comes to started blockchains is if you have these 64 different parts of your blockchain say one transaction needs to make it from one side to the other part of state and you need to make a transaction from A to B from one shard to the other, to be able to maintain that and make that transaction happen is a huge distributed systems problem in itself, it calls upon the concept of data availability which is this whole another concept when it comes to blockchains. Let's say I'm able to query the state of my shard properly am I getting the proper live version of the state of this shard if I'm coming in asking from another shard?

if we think about just taking over a singular shard which is a fraction sometimes one percent of the blockchain and say you do that, say you take over a single shard of this blockchain, if a bunch of these other shards are trying to communicate with you, you can cut off communication with them, you can lie to them, you can do all sorts of things to just disrupt the blockchain itself and if you're not implementing things like crosstalk communication and behavior and incentive alignment properly, single shot takeovers can absolutely freeze a blockchain. it's happened in the past and it's been pretty crazy so a lot of considerations when it comes to started blockchains but the promise of scaling is huge so if we go back to our little

triangle is a great example of how balancing things on this triangle, with sharding you are increasing scalability significantly but then when you go to something like decentralization you get a little more worried because now you split the blockchain so small that you might be able to take over and still have an impact on the blockchain. security guarantees are generally the same when it comes to sharded blockchains as long as you're implementing it properly otherwise you there is no reason you would want to go to sharding, it's just a little more difficult in security but you're still maintaining those guarantees.

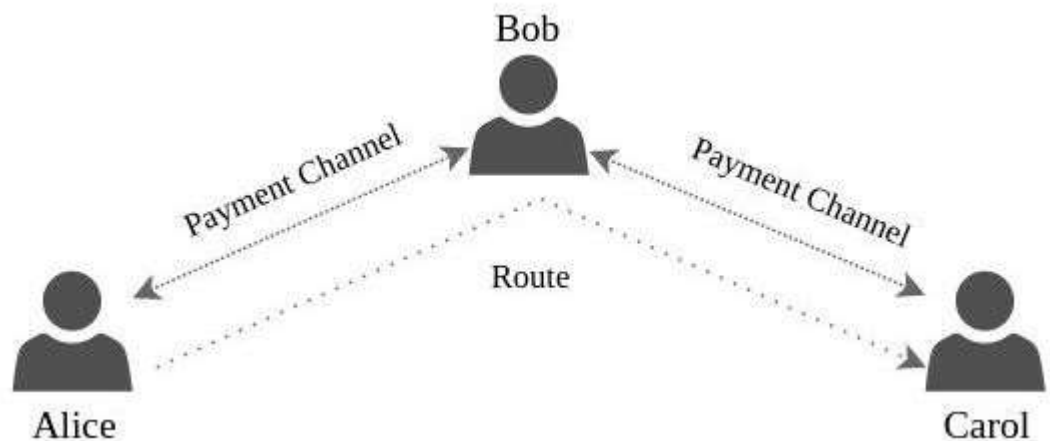
Layer two

What is Payment Channel Network?

A Payment Channel Network (PCN) is a type of blockchain-based system that allows for fast and efficient transactions between two parties without the need for each transaction to be recorded on the blockchain. This allows for greater scalability and lower transaction fees compared to traditional blockchain transactions.

- **Off-chain transaction:** A PCN typically consists of a network of payment channels, each of which connects two parties. These channels can be thought of as a form of “off-chain” transaction, as the parties can conduct multiple transactions between each other without each one being recorded on the blockchain. Once the parties are finished conducting transactions, they can close the channel and the final balance is recorded on the blockchain.
- **Multi-hop payment channels:** One key feature of a PCN is that it allows for the creation of “multi-hop” payment channels, where a transaction can pass through multiple channels before

reaching its final destination. This allows for more efficient use of resources, as the total number of channels required is reduced. Additionally, this makes it possible for parties to transact with each other even if they do not have a direct payment channel open.



In a payment channel network, Alice can transfer money to Carol by using intermediate nodes

- **Atomic swaps:** Another important aspect of PCNs is that they can be used to facilitate “atomic swaps,” which allow for the exchange of one cryptocurrency for another without the need for a trusted third party. This can be done by creating a payment channel between the two parties and using smart contract functionality to ensure that the exchange takes place as agreed upon.
- **Improves scalability:** PCNs have the potential to greatly improve the scalability and efficiency of blockchain-based transactions.

However, they also come with some challenges, such as the need for more sophisticated technical infrastructure and the risk of channel counterparty default. Despite these challenges, many projects are currently working to develop and implement PCLs, and it is likely that

they will play an increasingly important role in the future of blockchainbased payments.

How do Payment Channels Work?

Payment channels work by creating a locked fund between two parties. Let's consider an example of Alice and Bob.

- Alice and Bob commit an equal amount of cryptocurrency, for example, 1 BTC, into a multi-signature address.
- To release the funds, both parties must sign the transaction. This creates a payment channel where both can make transactions without broadcasting them to the network.
- Alice can sign a transaction that reduces Alice's balance and increases Bob's balance, for example, reducing Alice's balance to 0.9 BTC and increasing Bob's to 1.1 BTC.
- These transactions are held between Alice and Bob and can be updated multiple times.
- Bob can also sign a transaction to change the balance and send it back to Alice.
- This process can continue with as many transactions as desired.

Lightning Networks in PCN

Lightning Network is based on Payment Channels. The Lightning Network is a second-layer payment protocol that operates on top of a blockchain, most commonly the Bitcoin blockchain.

- It allows for faster and cheaper transactions by creating a network of payment channels between users.
- These channels allow for multiple transactions to occur offchain, reducing the need for each transaction to be recorded on the blockchain.
- This reduces the strain on the blockchain network and allows for faster, cheaper transactions.
- Additionally, the network of channels allows for payments to be routed through multiple channels, allowing for greater flexibility and accessibility for users who do not have direct channels set

up. It can also be used for other altcoins that support smart contracts.

Benefits of PCN

A payment channel network (PCN) has several advantages:

- **Scalability:** PCNs allow for off-chain transactions, which reduces the load on the blockchain and allows for a larger number of transactions to be processed.
- **Lower fees:** Since transactions are off-chain, fees are typically lower than on-chain transactions.
- **Privacy:** PCNs can provide more privacy for users, as transactions are not recorded on the blockchain and may not be visible to third parties.
- **Speed:** Transactions on a PCN can be processed much faster than on-chain transactions, as they do not need to be added to a block and confirmed by the network.
- **Increased security:** PCNs can be secured by smart contracts, which can ensure that transactions are executed correctly and that funds are not stolen.
- **Micro-Payments:** Because of low transaction fees and fast confirmation, PCN is well suited for micropayments.

Limitations of PCL

A payment channel network (PCN) also has several limitations:

- **Limited functionality:** PCNs are typically limited to simple payment transactions and may not support other types of transactions or smart contract functionality.
- **Complexity:** PCNs can be complex to set up and use, and may require a high degree of technical knowledge to implement.
- **Limited accessibility:** PCNs may not be accessible to all users, as they may require specific software or hardware to use.
- **Limited interoperability:** PCNs may not be interoperable with other blockchain networks or off-chain systems, which can limit their utility.

- **Centralization risk:** PCNs often rely on a central intermediary to facilitate transactions, which can create a centralization risk and increase the risk of a single point of failure.
- **Limited Liquidity:** PCNs are often limited to a small number of participants, which may limit the liquidity of the network.
- **Limited regulatory oversight:** Because PCNs operate outside of traditional financial systems, they may be subject to limited regulatory oversight, which can increase the risk of fraud or other illegal activities.

Conclusion

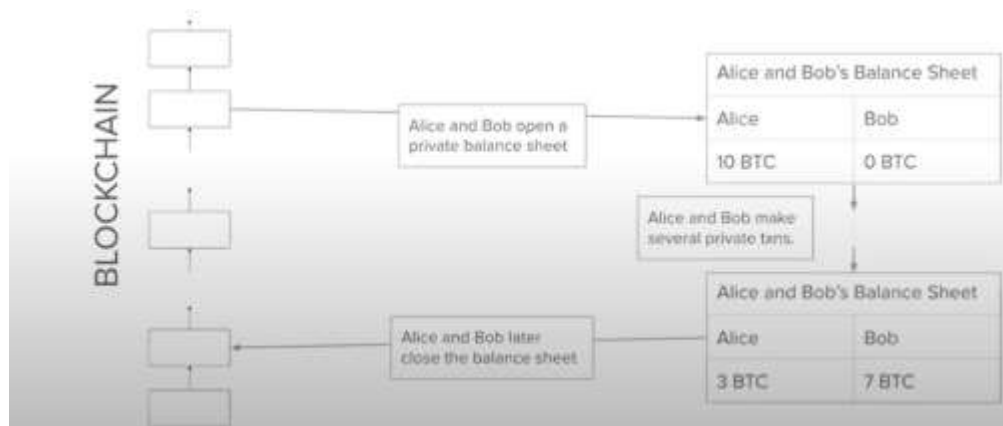
Payment channels and the Lightning Network offer a powerful solution for scaling blockchain transactions and making them faster and cheaper. By allowing for off-chain transactions and routing payments through multiple channels, the network reduces strain on the blockchain and enables greater accessibility for users. It is a promising layer 2 solution for blockchain networks and can be applied to not just Bitcoin but also altcoins that support smart contracts. The implementation of payment channels and the Lightning Network has the potential to greatly enhance the usability and mainstream adoption of blockchain technology.

Before jumping to the lecture itself reading the box above will help you understand better.

let's recall that every transaction is put onto the blockchain this is what we've been taught from day one, that's how we're able to guarantee that transactions are fully able to be settled by the system, that's how we're able to know that our bitcoin has been successfully transferred from me to niche or niche to me and the classic example of using bitcoin is buying a cup of coffee and let's say you pull up to your local starbucks and you want your frappuccino and you want to pay in bitcoin, best practices dictate that when you pay with bitcoin, you want to wait for about six blocks after your block to guarantee that your transaction has been put onto the honest and correct blockchain and to guarantee that it's guaranteed that your transaction (your bitcoin) has gone through. recall that one block being added is about 10 minutes so six block confirmation time corresponds to about an hour wait time which means that by the time you've confirmed that your block has gone through your coffee is already cold and you don't really want to eat it and you've wasted an hour of your day. each transaction has a transaction fee, this transaction fee is what goes to the miners, it's what incentivizes them to choose your transaction out of the thousands and millions of other transactions that put into a block, these fees are very inconsistent, they vary with regards to the volume of transaction in the network or maybe a price spike or drop without any warning, it's also very not economical for low value items, let's say the fee is like 10 cents in USD when you convert it and you're doing a one dollar transaction maybe you want to pay your friend back in bitcoin well that's like already just 10% of your transaction which is super uneconomical and this makes micro payments impossible which is interesting because micro payments open the doors to many different types of interesting and unique use cases for bitcoin.

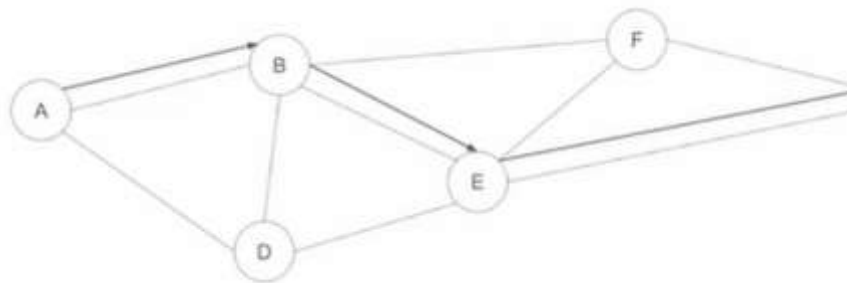
Let's go into how light network and fixes these or allows for improvements, there's an idea that do not put every transaction on the blockchain, you might be wondering if we don't put every transaction on a blockchain then what's the point of even using the blockchain? we're not going to take advantage of any of the guarantees, the security privacy and anonymity all those guarantees? this is where we introduce the idea of payment channels, let's say Alice and Bob open up a gaming channel between themselves and instead of consulting the blockchain every single time Alice pays him in bitcoin and Bob pays Alice in bitcoin assume that they're roommates and Alice paid for groceries one week and Bob paid for groceries of next week and they want to settle at the end of the month, what they would do is instead of consulting a blockchain, they have this payment channel open between them, they say like okay Alice has plus 10 bitcoin and groceries cost like eight bitcoin so then at the end of the month they need a result and Alice needs to have her eight bitcoin paid and he's had his eight bitcoin paid and they wouldn't have to consult the blockchain for two different transactions we would just consult the blockchain for the first transaction which opens up the payment channel. Let's say the payment channel is depositing Alice's 10 bitcoin and Bob's eight bitcoin and then the final transaction which settles right after they've calculated the net change in both of their balances, the final transaction would be like you have this much to save. that's the idea of a payment channel and an analogy is like when you enter a casino and you meet the man at the front and he's and you give him your USD in exchange for chips, you can think of that as opening a payment channel, you're opening a payment channel between you and the casino, then you enter the casino with your handful of chips

and you go to all the games when you're playing in the casino you're not dealing with USD you're dealing in chips which makes it very simple because you can think of that as on our private balance sheet we're just plusing bitcoin and minusing bitcoin, we don't need to consult the blockchain we don't need a deal and USD every time we make a transaction and then at the end of the day when we we're done playing games and we're either up a bunch or down a bunch then we'll go back to the guy the front of the door would be like here's our chips please give me back the USD, you can think of that as the final transaction in a payment channel that settles, after you're done gaining all your money then you change it back to USD and you walk out of the casino with a net positive or negative and you can think of payment channels in the same way.



this isn't really a new idea and payment channels have existed but what light network does is that it creates a network of payment channels. what payment channels allow is that we decrease the number of transactions let's say Alice and Bob have like 100 transactions what lighting network and what payment channels would allow them to do is only have two on-chain transactions and then 98 intermediary transactions aren't on chain. arbitrary or even infinite amount of transactions in between those truths

and trade transactions. what are those two transactions? well it's the opening transaction and the closing transaction and the in terms of the analogy it's when you enter the casino and you exchange USD for chips and then it's when you leave the casino and exchange chips for USD. the huge thing with lightning network is not only are you able to open up the payment channel with one person but the light network connects all of its users that are part of the lightning network in with payment channels with each other so you can imagine like we have Alice here we got Bob, Dylan, Ethan, Frank and Chris.



Alice and bob have a payment channel, Alice and Dylan have a payment channel, imagine nodes as users and edges as payment channels, what if Alice wants to pay Chris for something? well it would be kind of annoying if you enter a store and you're like hey I want to buy this thing and then you're like okay let's have a payment channel with the cashier and then you send bitcoin then you wait for the bitcoin to come back or whatever.

this is what lightning network does, it's much easier to just use the payment channels that already exist, like Alice has a payment channel with Bob, Bob has a payment channel with Ethan and Ethan has a payment channel with Chris and so why not just Alice can send some money to Bob then Bob sends that same amount of money to Ethan then sends that to Chris and then at the end of the day Chris has received the money that

Alice was supposed to send Chris. as long as there's a path in between two nodes then a transfer of money can exist that doesn't have to be on chain this can be off chain which means that you don't need to pay transaction fees, you don't need to wait for the six blocks confirmation time, you don't need to do all these things that all required when you are on chain you just need to use a lightning network and you know make sure you have that opening transaction that is on chain, keep in mind to open up the payment channel and then the closing transaction to settle all balances to whoever you paid throughout the duration of your stay in the lightning network.

Again we'll go into pros and cons to evaluate the lightning network. assuming that (keep in mind now we're not talking about the bitcoin network but we're talking about the lightning network which is remember in I2 solution so it's built on top of the bitcoin blockchain) there's enough money going around and everyone can spend money and send money to other users then people can make payments instantly because all we're doing is exchanging chip or we're just plusing to a balance sheet and minusing from another balance sheet, these are instantly compared to the six blocks confirmation time that is correlated with that corresponds with on-chain transactions you don't need to wait for confirmation times, you only use the actual bitcoin blockchain (you only use that base layer) to settle disputes and close out payment channels which is super efficient, now you have two on-chain transactions and in between those transactions infinitely (many arbitrary amount of transactions) and so now three transactions per seconds becomes how many traffic per seconds that's comparable to visa, paypal, mastercard and whoever. lightning network could truly be the thing that makes bitcoin usable on a daily basis. there are some cons with lightning network that happen.

number one is that these nodes or these users that are part of the network if we go back to Alice, Bob, Ethan, Chris, Frank, Dylan they all have to keep a decent amount of money locked up into like network so when you have bitcoin locked up in a light network then you can't use it elsewhere because the lightning network needs to use it as liquidity to spend (to send money to different users of the network) an argument can be made that all that money that's sitting there in lightning network, it's a waste of money. in addition to that there is a risk of centralization so with the scalability increase there is a pretty significant centralization risk imagine that one node (for example Ethan) is rich.

he has you know all the bitcoin and he can actually afford to keep his payment channels open way longer than anyone who has you know a small amount of bitcoin, he can afford to accept payments and route payments to where they're supposed to go much longer than anyone else and what this can lead to is this hub and spoke topology where this node in the center is Ethan and everyone else in the network is connected to Ethan in some way or another. what if an attacker builds up all these connections with other nodes but then all of a sudden takes away all liquidity, then there would be a cut in the graph and the network wouldn't be as connected as it used to which could lead to latency issues or payment errors and stuff like that.



centralization in this case is a very big risk but there are ways that lightning network is working towards to prevent this type of centralization absolutely.

Now let's talk about L2 scalable scaling solutions in the smart contract environment. when it comes to something like the lightning network it's particularly for things where you're working with a single cryptocurrency and you're working with payments specifically but when you go to something like ethereum are just on a different level because with ethereum you're now working with an execution context, you can execute any form of code on a blockchain and now with these smart contract execution environments you can't just create a channel between two people or just two other endpoints on the network you now need to bring an entire smart contractor (this executable context) onto the layer two so how do you do that and a popular modern solution to do that are roll-ups.

we'll highlight in particular a version of roll-ups called optimistic roll-ups. what a roll-up essentially is a contract on the blockchain let's start with Ethereum, a contract on Ethereum that is anchored directly to the Ethereum blockchain so similar to the lightning network with that casino metaphor you're entering and exiting a roll-up with transactions on Ethereum and then what happens happens in the middle in this time it's smart contract functions being called. this is now happening within the roll-ups context so you're starting and you're entering a roll-up, doing stuff on the roll-up and you're making changes to state that will then be saved in the end, once you get back onto Ethereum how do we make sure that these changes to state are legal, proper and approved and the

way that we do that we manage security in general with rollups are these proofs so with optimistic roll-ops we use things called fraud proofs and the way fraud trips work are when you exit the roll up itself there is a one to two week period in which anyone can challenge the roll up itself and be like I think you're wrong (I think the state that you ended up with is wrong and it shouldn't be like that) and then it's up to the roll-up contract itself to come out with this proof and be like no I was told to do x and I have this proof that I executed exactly that and then you will verify that proof and through that you'll be verifying the correctness of this off-chain computation.

that's the way that these roll-ups work in general then once the proof deadline has been crossed then everything is back to being finalized on Ethereum so the entire timeline is sort of create a roll up, create a relationship with your roll-up contract, execute a bunch of work with it as much as you want in a very cheap way because now you're no longer making transactions on Ethereum and then exiting and using this time period to submit proofs and verify state. what is great about rollups is that they are out and implemented right now for things like optimistic roll-ups there are a few implementations out there in the world. there's also a second form of roll-ups called zero knowledge roll-ups and these use rather than fraud proofs to validate rollup state zero knowledge proofs. these really often computationally expensive so that's why they're not out yet but extremely fast in terms of finality Roll-ups (by mentioning finality it means that for zk roll ups you no longer have this one week period of challenge which is much better when you think about it, you enter the roll up and when you exit you know that you're fully back on Ethereum) so zk roll ups in the long term are definitely the move but for right now the optimistic roll-ups are the ones that have been able to be deployed.

these roll-ups offer a fair deal of scalability often 300 to 400 times in terms of gas costs which is pretty awesome so you're able to handle much more transactions in a transactions load on the layer two but of course on the other side of the scalability triangle you have to worry quite a bit about security and that's why you might have to make trade-offs in terms of one to two week challenge period to get back to finality on Ethereum. decentralization is again a little orthogonal, nothing too much is happening there.

Lecture 9

Decentralized finance(Defi)

<https://www.youtube.com/watch?v=XzBrZq7jib4&list=PLLkiRvMHnp6OIWkZVa85g2tv7NtlgoAID>

Before going to the main lecture read the box below for better understanding (it's totally optional, you can go to the box if you didn't completely understand)

What are stablecoins? Stablecoins are a category of cryptocurrency whose value is pegged to another asset, most typically a fiat currency, to maintain a stable value in the ultra-volatile world of crypto. Stablecoins are typically designed to maintain a set ratio to an underlying fiat currency to which they are pegged. The most common peg is the United States Dollar ("USD").

There are three key characteristics that define a stablecoin: stability, capital efficiency, and decentralization. As of today, any given stablecoin falls on a spectrum of meeting these three characteristics. Typically, only two characteristics are met. The market has not seen a stablecoin successfully integrate all three characteristics simultaneously. This is known as the stablecoin trilemma.

There are many varieties of stablecoins on the market, but the three most common types are fiat-backed, crypto collateralized and algorithmic.

First, the most well-known and widely adopted stablecoin in the market is the fiat-backed stablecoin. Fiat-backed stablecoins are digital assets that maintain financial reserves in fiat currency held by a regulated institution such as a bank. This type of stablecoin holds its reserves in a bank vault or with a trusted financial custodian. Fiat-backed stablecoin reserves are a weighted mix of cash and cash equivalents such as commercial paper. For example, a \$10 billion fiat-backed stablecoin may hold \$4.0 billion in physical cash and the remaining \$6.0 billion in cash equivalents. The centralized entities operating this type of stablecoin must generate revenue and do so through the yield on their cash equivalents. The two largest stablecoins by market capitalization,

Tether & USD Coin (USDC), fall in this category. As previously mentioned, these

stablecoins are not decentralized; however, they are stable and capital efficient, thus allowing for significant scaling. Their stability is reliant upon the entity maintaining significant reserves coupled with compliance with both auditors and regulators to prove transparency of these reserves.

Second, crypto collateralized stablecoins are a popular variety that reside exclusively on the blockchain. This type of stablecoin collateralizes crypto assets via an electronic vault system. The collateralized vault triggers a smart contract to mint brand new stablecoins. Due to crypto's inherent volatility, assets are often overcollateralized with ratios routinely reaching 200%. As a result, this type of stablecoin is highly decentralized yet capital inefficient. A crypto collateralized stablecoin gives up capital efficiency to maintain decentralization. As the crypto market matures and volatility decreases it is likely the collateralization ratios will fall. The overcollateralization structure is a form of stability protection designed to keep the stablecoin from depegging (losing their desired 1:1 ratio). Over time, the market capitalization of these decentralized stablecoins should continue a slow, but methodical growth.

Third, algorithmic stablecoins are a newer category that also reside exclusively on the blockchain. This type of stablecoin relies upon an algorithm that pegs itself to a physical currency. They are not fully backed by collateral and rely upon an algorithm to maintain their 1:1 price peg. The laws of supply and demand are integral to the mechanics of this variety of stablecoin. When demand for the stablecoin increases and its price subsequently appreciates over \$1.00, the protocol issues a new supply of stablecoins to drop the price back to its pegged \$1.00 price. Alternatively, when the price of the stablecoin falls below \$1.00, stablecoins are removed from circulation through a process known as burning. Due to its location on the blockchain as well as its undercollateralized nature, this stablecoin is both decentralized and capital efficient. The weakness of these stablecoins is a lack of stability. Stability is a highly desired feature of stablecoins, and it has proven an elusive goal for all

algorithmic stablecoins to date. Earlier this year the algorithmic stablecoin TerraUSD (UST), lost its peg. This depegging caused a quick collapse that severely impacted the entire crypto market.

It is not clear today which stablecoin model or models will prevail. As innovation continues to drive forward the crypto landscape, we expect to see continued efforts to solve the stablecoin trilemma and solidify the position of stablecoins as an integral pillar of the crypto economy.

Investing in cryptocurrencies comes with risk that may not be suitable for all investors, and as such, it is vital that potential investors conduct their own due diligence before buying or selling any cryptocurrency.

<https://www.elliottdavis.com/stablecoins-fiat-backed-vs-crypto-collateralized-vs-algorithmic/#:~:text=Fiat%20backed%20stablecoins%20are%20digital,with%20a%20trusted%20financial%20custodian.>

What is Defi?

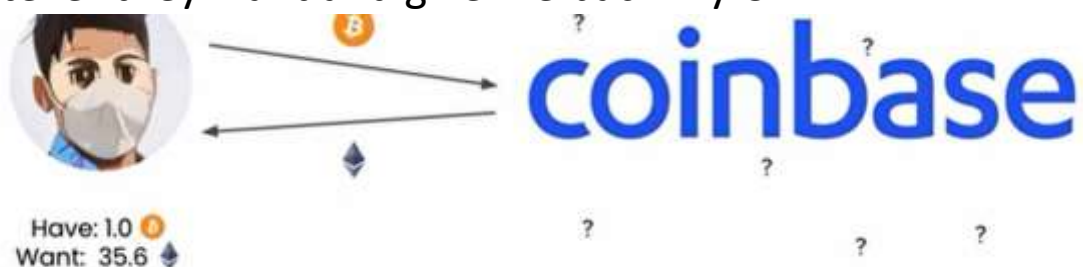
when you ask yourself or if you ask someone what defi is or if you just search it up online you might run into many things, you might run into a bunch of scary big numbers about percent returns or a bunch of logos and a bunch of food based DeFi protocols, you will might also get a little bit overwhelmed with how much there is around there because you will run into hundreds of projects that call themselves DeFi companies so it can be a little overwhelming.

If ask ourselves what is DeFi really without the flaw for speculation or degeneracy is a common qualifier? to many, DeFi is the movement to build an open, global and

interconnected financial system without the drawbacks of traditional finance we've discussed. in something like DeFi we have an open financial system where you are not stopped by any central authorities who are here to change or manipulate or control the way that you trade.

finance is a little more complicated than just cryptocurrencies, we have cryptocurrencies already which are great but what else can we do with these cryptocurrencies and financial systems? financial products are the kind of answer, these complex trading platforms or lending platforms etc are what comprise DeFi about. financial applications and power to the people as I like to call it DeFi on a concrete level are often applications built on smart contract blockchains like Ethereum. Ethereum smart contracts are the mode that most DeFi applications right now are deployed on and DeFi is also continuously developing it to many people, right now is in a very experimental phase (we're only in the first few years of figuring out what DeFi means) so right now to many people it's an experiment that has a lot of room for growth.

There is another sort of motivating example that is highlighting the decentralized part of decentralized finance, when you think about it and you ask yourself is every blockchain based financial tool decentralized? well no not exactly. let's take something like coinbase if I hold a bitcoin and I go to coinbase and I'm like hey I have bitcoin and it's also my coinbase wallet, I would like to swap it, if I need to do that I send coinbase over my bitcoin they do whatever they want and give me back my UTH.



this is weird in a few ways, you are really truly trusting coinbase with your money in this and you could give them your bitcoin and they could do whatever they want with it, they can put whatever price they want with it and in the end it doesn't really mean much, all they have really at stake from hurting you is just a reputation, what if coinbase gets breached, you're absolutely dead. what we want is decentralized platforms where what you see and you know exactly what to expect because a lot of these applications are on these smart contracts where you see either logic fully mapped out or you know exactly what calls are going to be made and things are very properly auto dated and public.

Pillars of DeFi why do we need stable coins? and what are stable coins? a lot of times people's first complaint when you mention cryptocurrency as a solution to something (as a response to our current centralized financial systems) is the volatility of cryptocurrencies and if I want a currency that maintains a stable value.



if we think about purely monetary stability, then stable coins are kind of a response to this, it's pretty obvious from the name itself but the goal of a stable coin is to be a cryptocurrency that maintains a stable value oftentimes.

there are many types of stable coins, the first is fiat backed which is often sorta-centralized, this means that in order to generate or mint Uther stable coin you have to drop down a certain amount, often an equal amount of fiat currencies like dollars or euros or pounds, and you'll go to one of these portals and drop down your fiat money and then an institution often, for something like USDC the circle institution or USDT the tether organization, will then issue you the stablecoin and because they're issued by these organizations while they still are stable coins a bit of their own functionality is controlled by these organizations themselves. they're very strong stable coins but because they have this little bit of centralization to it, where these organizations can maintain these pegs very sharply which is a little worrying so we have these fiat-backed sortacentralized stable coins which are still pretty rampant around DeFi right now. the proper wave and the things that people really get excited about are crypto-backed decentralized stable coins, now you're no longer even using fiat currencies to generate your stable coin, you're just using cryptocurrency themselves to create stable coins that can't be controlled by institution, in the ways that we've seen these other fiat-back stable coins. in the long term we more care about these decentralized stable coins.

Dai (or **DAI**, formerly **Sai** or **SAI**) is a stablecoin on the Ethereum blockchain whose value is kept as close to one United States dollar as possible through a system of decentralized participants incentivized by smart contracts to perform maintenance and governance functions. Dai is maintained and regulated by MakerDAO, a decentralized autonomous organization (DAO) composed of the owners of its governance token, MKR, who may propose and vote on changes to certain parameters in its smart contracts in order to ensure the stability of Dai.

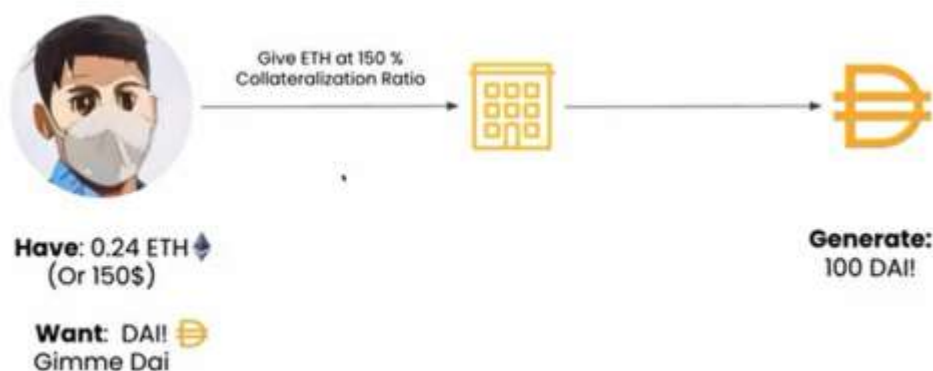
Together, Dai and MakerDAO are considered the first example of decentralized finance to receive significant adoption.

[https://en.wikipedia.org/wiki/Dai_\(cryptocurrency\)](https://en.wikipedia.org/wiki/Dai_(cryptocurrency))

we're going to talk about the ultimate decentralized stablecoin right now which is DAI, the premise of DAI is that we have this goal to create a stablecoin and we need to do this using cryptocurrencies, how do we do that if these cryptocurrencies that we're going to use as collateral to generate our stablecoin are volatile themselves? We should put in more volatile crypto than the stable amount that we want, let's over collateralize right now, there is a collateralization rate that gives us what percentage or what ratio of collateral do you need to get an x amount of stable coin back, so if you do that you will get the ratio which is more than one.

The other question is why would I this, why would I put more money into this vault? I'm losing money. there is value in having a stable currency itself that is enough to be worth it.

Now let's see how this works.



we have me on the left and I have around 150 dollars in eath and I want DAI, then this little building in the middle we'll call that the "DAI smart contract" so these are smart contracts on a blockchain and I'm making this call saying hey I have eth can

you generate DAI for me? and if I give you ETH at collateralization ratio of 150 meaning to get x amount of DAI you have to pay 1.5 times that in your collateral, I will then get 100 DAI back and this is like the DAI smart contract is loaning this out to me then placing this ether as collateral and what I have now is this DAI and an outstanding loan and I have this debt to the smart contract and eventually when I'm done doing whatever I want with my DAI, I need to give it back to the DAI smart contract in order to get my collateral back and there you go, you've done what you want to do perhaps you made a little bit more money. is it really that simple?

let's talk about the risks here, the big risk here is the volatility of the collateral that you put in, when you have these volatile cryptocurrencies that you're using to collateralize your DAI there's always the chance that if you put in today drops in value such that it is worth less than 150 percent of the DAI that you borrowed and that there is a fragile position because the value of DAI isn't stable right now, it's not being backed by how much it expects to be backed and the response that the DAI smart contracts do to this is doing something called a liquidation what happens is that the guy contracts will do everything to get that DAI back into the system because that's what the debt is right now, you owe them DAI and they need to get it back at any cost so they have your collateral already what they're going to do is then sell your collateral to others for DAI and in the end because you are over collateralizing you might not lose all your money but you will lose that 100 worth of your collateral and it's gone because you did not manage your position properly is the stance of the DAI contracts. there are many others fail-safes but liquidations are the main response to the volatility of cryptocurrencies and how they can change stable point positions.

Decentralized exchanges

so once again the what and why of DEXes?

it's a very clear use case to have something like a decentralized exchange but we'll split it up into two parts. first off why do we need exchanges for cryptocurrencies? this one's pretty obvious, with so many cryptocurrencies out there how do you go from A to B in terms of the cryptocurrency if I have eth and I want to get something like DAI do I have to only go through maker contracts and generate there? is there another way to get it? you have exchanges to go between different cryptocurrencies like you might between the pound and the dollar.

Second is Why decentralized? this goes into the coinbase example, remember when we had coinbase where we could have coinbase execute trades for us but it would just be this little black box that we really wouldn't know what was happening around and the people at coinbase could make decisions to completely change the way that my trading experience was, we want to give power to the people and to the markets by having everything executed on these public smart contracts and that is where the decentralized manner comes in all shapes and sizes.

There are two general categories of DEXes to highlight.

first off is the order book this will hearken back to how you know exchanges just in the real world, the centralized finance world, and there are open orders on the market, there are many different terminologies but there's like the maker orders and the tape orders where there are people who are like I want to buy a currency for x and someone will say I want to sell x amount of currency for x and then they kind of match up

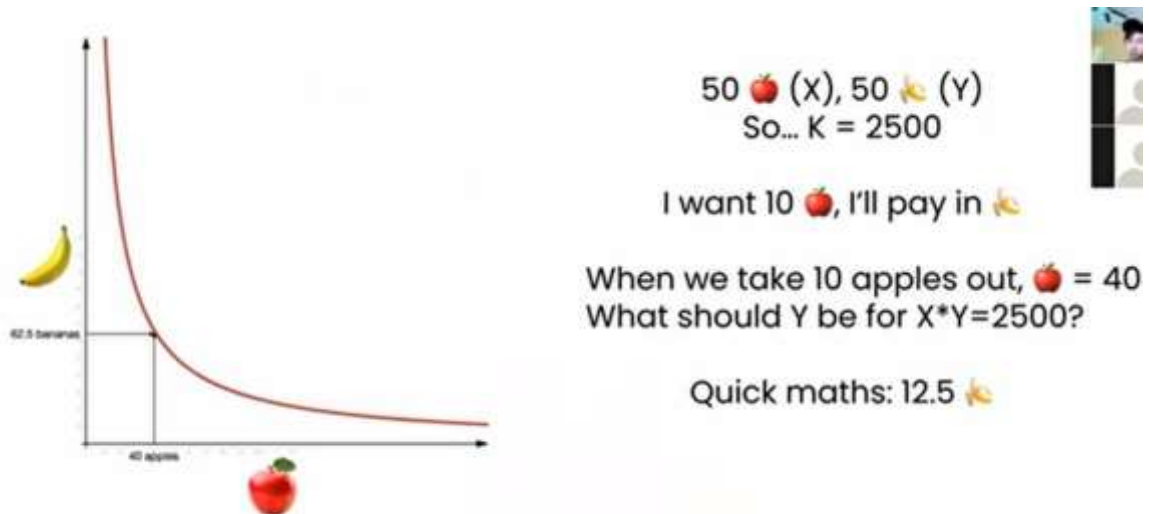
together using a little algorithm. Another exchange is the automated market maker, this is a new sort of exchange system that has become very popular in DeFi right now. So let's talk about it.

automated market makers mean that the price that you are quoted and get for making a trade is fully based off a function of the supply in these liquidity pools.

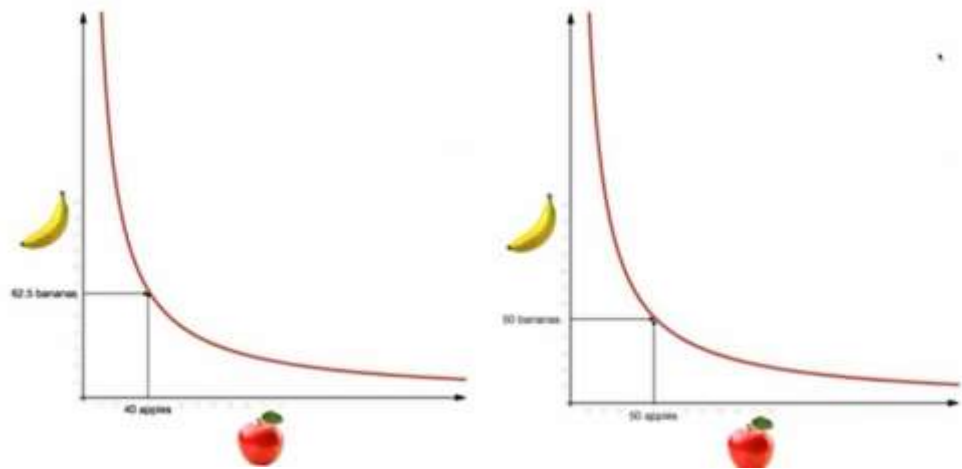
let's start with this liquidity pool concept, rather than having orders and a list of orders to create this order book why don't we use a pool of liquidity where we have all the suppliers, supplying their cryptocurrency together into a pool and then why don't we go in and out of that pool rather than matching this particular order. there are a few distinct advantages using liquidity pools and having function based market makers. for us what's really interesting is that it handles low liquidity situations and that's great because when we're working with things like cryptocurrencies, volume and liquidity are very volatile factors because in something like an order book method when you have low liquidity you have also a low amount of orders, a low amount of takers on your orders and when you have that this kind of unideal and slippage becomes very high slipped which is the difference between the price that you are being quoted and the price that the trade eventually executes at but AMMs are very adaptable and they'll be very nice low liquidity situations and in the very fast-moving world of crypto right now are very useful.

so let's say we have a pool um for trading Ethereum and DAI together so we have this eth-DAI pole how do we get a price for this? oftentimes the method that people go for are these constant product functions ($x \times y = k$), we have this function where x and y are the supplies of the cryptocurrencies in the pool themselves. let's see how this

works and we're going to go through the lens of something like Uniswap, a very awesome decentralized exchange one of the first to centralize these changes. so we're going to go through a nice little example, in our pool we are swapping between bananas and apples, apples and bananas and in here apples will be our x and bananas will be our y and the x and y here are supply numbers so we have 50 apples and 50 bananas so k is 2500.



if I know that I'm gonna have to put in some amount of bananas in order to get 10 apples out and the way the constant product works is the function basically asks itself if we take 10 apples out of this pool we want k still be 2500 so what should y be? for x times y equals 2500 if x is 40, with a little bit of quick maths we get 12.5 bananas so I will be quoted at 12.5 bananas in theory and we can kind of see this happen on the diagram.



that is how a constant product automated market maker works a lot of a lot of terms there. there was a little bit of a cambrian explosion in DEXes early last year we saw a bunch of DEXes that built upon the classic Uniswap, AMM model.

we can start with things like balancer where their main focus is generalizing everything, in Uniswap pools are 50 50 weighted meaning that the amount of cryptocurrency that you supply in has to be equal to the amount of eth in the pool, it is also the same as the amount of DAI balancer. we can actually allow for an arbitrary weight here so we allow people to create tools with arbitrary weights, arbitrary fees, arbitrary whatever you design your aiming on how you want.

curve is a very cool DEXes, it's very stable coin centered and it uses the same sort of AMM features that from before but instead with a nice price function that is tailored towards trading stable coins so things like USDC, USDT, DAI, USD, BUSD. Bancor: their direction is in tackling and permanent loss which is a very interesting concept but you can understand it as a bit of risk that providers or lp's liquidity providers in these AMM pools have to go through and they tackled that by having both a native token and also allowing you to do things like single token exposure with your provider only having to provide a

single token or other things like that. they kind of tackle the risk side of things



Generalize Everything!
Arbitrary weights, fees



Stablecoin-based DEX with
a fancy new price function



Tackle **-Impermanent
Loss-** with a Native Token +
more

Lending markets

say you have these cryptocurrencies and perhaps you don't want to completely give up your exposure to play around with the other, like with something like a decentralized exchange so once again we have eth, DAI maybe I don't want to just have to swap my eth to get DAI what if I have my UTH, I want to maintain price exposure to it where there are spikes, while still being able to play around with another cryptocurrency like DAI and that's what lending and markets are for especially borrowers. they give you that purpose in exposure to other cryptocurrencies and for lenders, you also get a very nice way a trustless way to earn interest and get access to these interest rates without central points of failure.



Purpose

Decentralized platforms where you can accrue interest by locking up coins in smart contracts (no banks!) or issuing P2P loans which earn interest over time.



Reasoning

Trustless way to earn interest and get access to market efficient rates without reliance on a central point of failure.



Value

Increased access to financial interest on assets with personal collateral without sacrificing your identity or financial history.

Let's start by a couple of examples to show the diversity of the lending market.



Lend and borrow
crypto to earn stable
and variable interest.

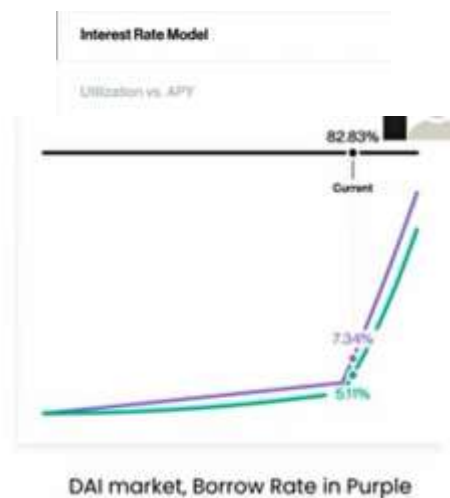


Multi-Collateral Dai
(MCD) enabling users
to generate Dai by
staking collateral
assets.



Algorithmic,
autonomous interest
rate protocol built for
developers to unlock a
universe of open
financial applications.

The how of DeFi lending in a lending market for a specific asset let's say that we are looking at the lending (meaning the supplying) and borrowing of DAI, these all revolve around interest rates because the suppliers have to be incentivized in some way to give their cryptocurrency to the protocol and then lend out to other people and interest rates are often set by a function of utilization, utilization being the percentage of the current market being loaned out currently. for example if we have 80 utilization meaning that 80 of all the crypto in this market, the stable market, is being loaned out right now.



as more is being loaned out, we do need to charge higher interest rates because it is a popular market and these interesting models will work in that way. as utilization goes up the rate that you pay also goes up. the kind of tagline is

borrowers pay interest on their crypto based off these interest rates which is distributed to suppliers equally. one kind of cool tidbit to understand is that the supply rate the, rate that suppliers get on their crypto, is always going to be a little bit less than the borrow rate and this is because the amount of cryptocurrency that is supplied is greater than the amount borrowed out at all times. you never really want 100 market so then you have to pay or distribute this paid interest by more borrowers or less borrowers to more suppliers and because of that you get a couple percentage smaller supply rates and these lending protocols often add their own additions to their lending markets.

Derivatives

before going into DeFi derivatives we're going to talk about derivatives in traditional finance the idea of a derivative basically is that you have this contract and this contract or this agreement has a price that is based on an underlying different asset, it derives its value from an underlying asset. if you're looking at something and you hear that this type of asset is a derivative you know that there is some kind of underlying asset that it derives its value from and these underlying assets are like stocks bonds, commodities, currencies, crypto as well and changes in the underlying value (the value of these underlying assets) is what changes in the value of the derivative and derivatives are mainly used for two reasons, in the traditional financial system those are hedging risk and for speculation. a derivative is something that derives its underlying value from something else and an option is a contract that gives you the option or gives you the choice to buy something at a certain price or sell something at a certain

price at and a certain date. an example of an option could be right now I could purchase the option to buy tesla stock at 600 up until the date

September 1st, this good for me in a case that let's tesla's stock price skyrockets to like 900 then on September 1st when I'm looking at tesla stock price 900 then I know that I've purchased this option that allows me to buy it at 600 and this option worth something because by using the options contract I have tesla stock at 600 and then sell it immediately at 900 and pocket that 300 per share.

Hedging is a strategy for reducing exposure to investment risk. An investor can hedge the risk of one investment by taking an offsetting position in another investment. The values of the offsetting investments should be inversely correlated.

how could this be useful for hedging risk? let's say you want to buy the option to sell tesla at six hundred dollars at any time up till September first and then you have a bunch of tesla stock and it plummets to like four hundred dollars if we reach September 1st we've purchased this option to sell tesla stock at 600 and we see that the tesla price is now 400 well that's great because if we want to exercise this option then we can we can save ourselves the grief of selling at 400 and sell at 600.

whenever a bunch of money flows into something really quick there's always a danger of a correction happening or a price retrace happening. people were worried what happens if the price of compound the governance token drops too far that's when open finance this is a company that was actually started by a blockchain at Berkeley open finance released these options contracts on their DeFi platform that basically gave you the right but not the obligation (I don't have to do it if I don't want to) but it gives you the right to sell, back then the price of

comp was around 200 or 250 and it gave you the right for about a couple weeks to sell the comp token at 150. this means that if you buy a bunch of compound token and then you also bought that option then even if compound drops to like 70 dollars (it drops really far down like below 150) you still have this contract that gives you the option of selling your compound tokens at 150. so you basically protected yourself from loss below 150. the cool thing about DeFi derivatives in particular is they're really lowering the barriers required to get involved in these sophisticated mature tools, usually in traditional finance, for accessing a lot of these tools you have to have access to many of these exchanges and even then for like things like margin you have to like get further vetted, so DeFi really provides access to many of these services and lowers barriers to them. Now let's talk about some of the top defy derivative platforms.

SYNTHETIX

Trading derivatives in DeFi and exposure to derivative assets.

$\delta Y / \delta X$

DEX, lending and leverage and a unique provider of derivatives in the DeFi trading market.

LMF

Open source protocol enabling two parties to create their own financial contracts that are secured with priceless tokens.

synthetics is a trading platform that allows you to trade derivatives and get exposure to a lot of different assets in crypto, for example one of the things that synthetics offered was synthetic gold, this is basically a crypto token that was a derivative and it derives its price for some kind of underlying asset and that underlying asset was gold so basically it moved up and down with the price of gold and this is really interesting

because if you've heard the narrative of bitcoin as digital gold the reason that people really try to sell bitcoin as digital gold is because the idea of using gold as a store of value is something that people have really understood for a long time, this idea that we need something that is true and that is scarce to store our value, and that's something people understand, people have bought into for a long time and what the problem with buying and owning gold is that there's a lot of barriers to entry involved for example if you're trying to buy physical gold where do you store it? how do you keep it safe?, etc. if you just buy the synthetic gold token well now you're the price of that token goes up and down with gold but you don't have to deal with all the hassle of actual gold. place is dydx, this is really interesting because it allowed access to perpetual futures which functions leverage, a perpetual future is a type of contract that tracks the price of whatever it is a perpetual future, so one perpetual future contract for bitcoin is the same price as bitcoin so it's kind of a way of buying and selling these perpetual future contracts in a way of getting leverage on DeFi or on your crypto assets without having to borrow from someone. dydx really allowed decentralized access to this. UMA is another super interesting one and it stands for *universal market access* and it enables DeFi developers to build their own synthetic assets so this basically allows any developer to create a contract between two parties and make their own derivatives and that really democratize a space.

DeFi Asset management

this is similar to derivatives in that they're kind of a tool that had limited access in traditional finance, that was that really like in DeFi had its barriers lowered and was seen as more accessible to a lot of different people, so asset management in DeFi has a lot of interesting standards that need to be discussed for example they have to be non-custodial, you want to make sure that the underlying assets is not revoked the cool thing about DeFi is that it's not really restricted by any human errors or human limitations so there's no concept of the exchange is closed at this time or I can't call my over-the-counter person to do this trade because it's closed right now or something like that.

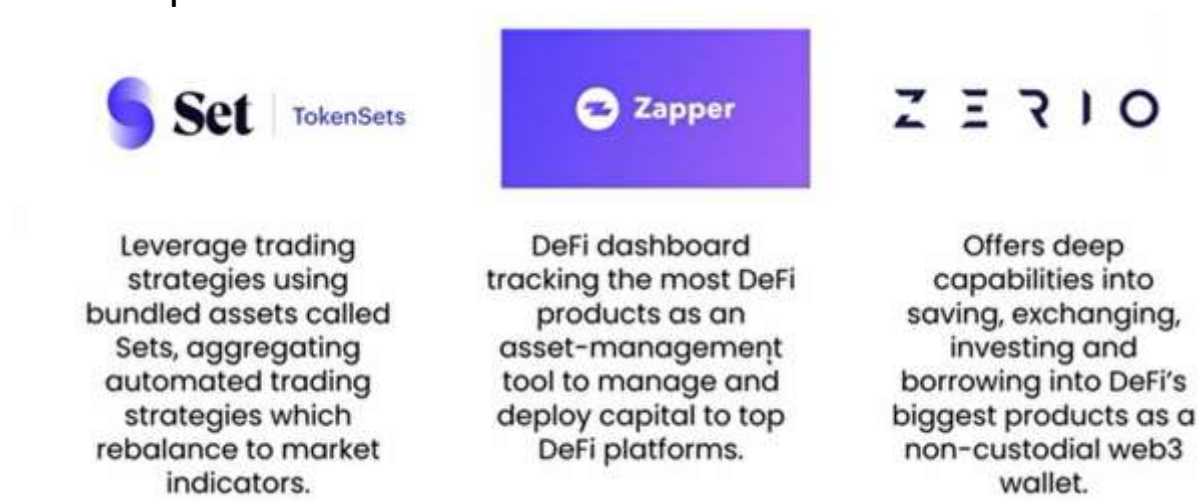
you want your asset management solutions to be composable, this means you want them to be connected to a wide number of DeFi projects and to be able to move your assets around in a lot of different places and remove friction and remove barriers, this really contrasts well to the traditional finance market where a lot of different derivatives or access to assets requires a lot of different accounts on different exchanges so I'll have Coinbase, Fidelity, etc to get access to a lot of different things, my stocks, my crypto, my Roth, my 401k stuff like that and DeFi asset management really does its best to reduce the number of platforms that you have to be interacting on.

you want these to be automated so a lot of things in DeFi asset management require active interactions such as rebalancing and liquidation and these basically happen without the user having to intervene.

you want these to be globally accessible, like crypto DeFi is without borders, there's no concept of you being in a certain country.

- **Non-Custodial**
 - Ownership of underlying assets never revoked
- **Composable**
 - Connect to a wide number of DeFi projects
- **Automated**
 - Rebalances and liquidation occur without user intervention
- **Globally Accessible**
 - Accessible to everyone regardless of income/location
- **Pseudo-Anonymous**
 - Connect through wallet address enabling anonymity

Let's me talk about some top asset management platforms in the DeFi space.



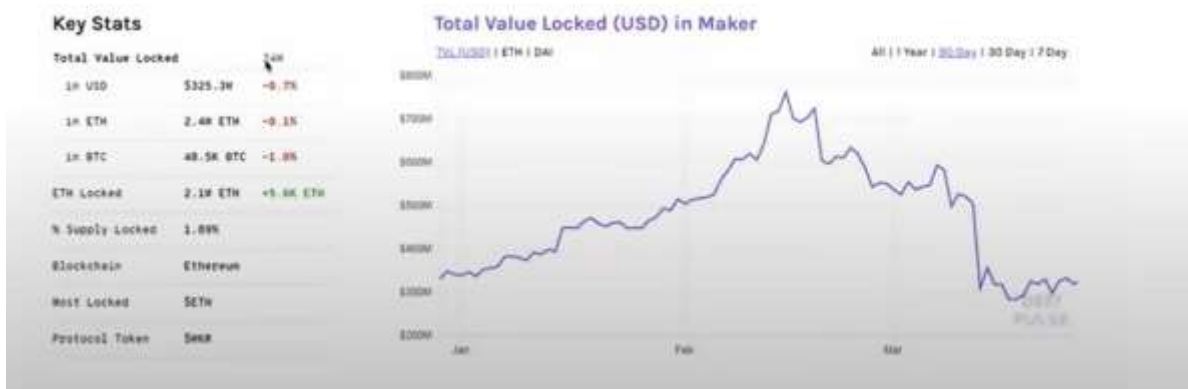
zapper and zerion are really similar they're basically like DeFi dashboards that basically allow you to track all of your DeFi investments and your liquidity, any places that you're providing liquidity and things like that all in a certain centralized location and set protocol is actually an automated way to get access to a basket of assets. what happens is that like aggregate automated trading strategies and rebalance to market indicator, let's say you want access to the crypto equivalent of an eth, protocols is where you can get access to these bundle assets that track different sectors of crypto and automatically rebalance kind of analogous to an index fund or managed funds and the cool thing about it is that it's decentralized and doesn't require access to some kind of exchange.

DeFi summer

now that we've discussed our pillars of DeFi and the tools that we've now been given access to, now let's talk about this series of events that people commonly refer to as the DeFi summer that really took place over the course of March through August, September of 2020 where a lot of projects came out with a lot of interesting things, there were hacks, there were air drops, there were various different things that basically shaped how DeFi all been and how it is today and kind of what we value in DeFi and all of this really started with the MakerDAO hack this really is what kicked off DeFi summer, this is back in early march, this is the organization that created the DAI stable coin that we discussed earlier and was one of the biggest players in the DeFi space and one of the most interesting and relevant projects especially at the time, so what happened is their system got hacked and the total value locked in maker and as a result the entire DeFi space dropped by a really large percentage and it was at a really inopportune time because it kind of opened up the playing field to a lot of different players because maker was really the biggest project in DeFi but it also kind of coincided a little bit at some level with the stock market crash that happened back in march. there was this whole event called black Thursday that was related to this, that saw drops in prices of a lot of crypto assets but on the other hand it really opened up the playing field to a lot of different players and many people took advantage of this.

The MakerDAO Hack

- 50% Drop in DeFi TVL Locked
- Loss of Trust in Large Protocols
- Necessity of Security in DeFi platform development



one of the interesting projects that followed this was balancer who released liquidity mining, this is basically another has an automated market maker system or an AMM system but this allowed users to provide liquidity and rewarded them with their own governance tokens, so now you're taking whatever a thousand, two thousand dollars of your money and you're providing liquidity to a lot of different pools uh one of the very notable and unique things about balancer is that when you're providing liquidity of the pools it's no longer 50 50. so usually when you're providing a liquidity pool let's say you're providing to the eth bit like the BTC pair, you'll divide your funds into half eth and half BTC and then provide to both pools but balancer really allowed you to provide a lot of these interesting splits.



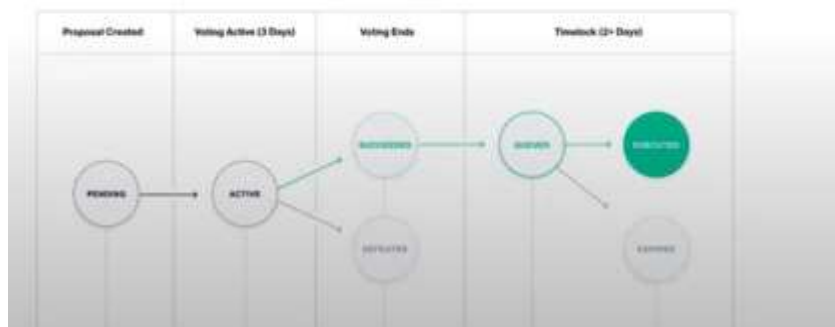
there was one that was full of stable coins and basically gave you a lot of different options in ways that you could provide liquidity and also incentivize people by giving them their

governance token. governance tokens, you can think of them almost like stock in these crypto companies, when you have a governance token you're basically allowed to based on how much you hold, have a say in the new developments, in this platform and this concept of being able to earn governance tokens on the platform where you're providing liquidity really was attractive to a lot of people because now you're participating in a system and you're getting incentives but you're also getting a say in how this platform moves forward and especially getting governance tokens by providing liquidity was a decentralized way of distributing these tokens and many projects followed soon after a compound released their governance token. this is a token that we were talking about earlier that open finance created an option for, but this governance token was basically distributed via liquidity mining also on the platform and it again allows users to be involved in how the platform moves forward. a quick description of how governance tokens work, governance token holders are allowed to make proposals on changes to the system or changes to the protocol and if there's someone else who agrees with those changes then they can stake their tokens on your proposal based on seeking there's a process of vetting, the process of approvals, and then people will vote on these proposals and based on that vote they will either be executed or not executed.

it's like taking decentralization to the next level, first you have the level of having a cryptocurrency, a currency that is decentralized, where the transaction verification is decentralized then you have the next level in which is creating platforms like decentralized exchanges which is where you're able to keep your tokens in decentralized custody solutions and then also trade them in a way that is decentralized and not

overseen by a company and where transactions are not verified by a central authority then you almost have the next level where those platforms and what decisions they make those decisions are also decentralized. this concept of distributing governance tokens and distributing the right to have a say on where these platforms go that kind of was like the next level of decentralization and people saw their to acquire this this governance token and have a say in how a platform does things.

It was really an attractive prospect that brought a lot of people compound their total value locked, skyrocketed or like doubled almost in three days after launching their governance tokens. this concept of governance tokens really was a contributing factor to how DeFi moved forward, one of the problems though was that there were a lot of large VC companies and firms that that did their best to acquire as much governance token as possible and then what kind of started happening was a lot of these governance proposals started being dictated by large companies which is the opposite of what we wanted to happen.



moving forward we have Yearn.Finance(YFI), Yearn.Finance was a project that was started by this developer named Andre Cronier, he has a decent amount of notoriety in the DeFi and the crypto space.

The saying that he goes by is I test and prod which basically means like I test my code in production, so I don't test my code first I deploy my code and when it's in production that's when maybe bugs will be seen uh and Yearn.Finance was really seen as a gateway to all of DeFi.

What Yearn.Finance allowed people to do was have the yield they earned from being a liquidity provider to these liquidity pools or to these AMM protocols to have that yield be optimized so for example if you want to provide to stable coin pools or some kind of liquidity pool with a certain token Yearn is gonna go around and do its best to find which pools are currently providing the best yield and then it would reallocate its assets every time someone would either withdraw or add to the pool and then you get a token that represents your stake in the pool uh and that could also be reinvested into other platforms.

Yearn was one of those projects that plugged everything together and it's definitely one of those connecting blocks in these money legos so people would put their money in a certain project or in Yearn and then get their tokens from Yearn, put that in somewhere else and do their best to maximize their yield and one of those projects was curve finance.

- Created by Andre **"I test in prod"** Cronje
- Gateway to all of DeFi
- Hold asset while earning yield
- Get token that **represents stake in pool** that can be reinvested into other platforms and **be exchanged (stakes of pools)**
- YFI, YFFI, YFII (Forks)

in the beginning of Yearn.Finance it gained a lot of traction and got hundreds of millions of dollars of value locked and people realized a couple days that based on this mark, the way that the

smart contract, Andre, had access to the value locked in Yearn, he could have just taken the hundreds of millions of dollars that were there.

What Is Slippage?

Slippage refers to the difference between the expected price of a trade and the price at which the trade is executed. Slippage can occur at any time but is most prevalent during periods of higher volatility when market orders are used. It can also occur when a large order is executed but there isn't enough volume at the chosen price to maintain the current bid/ask spread.

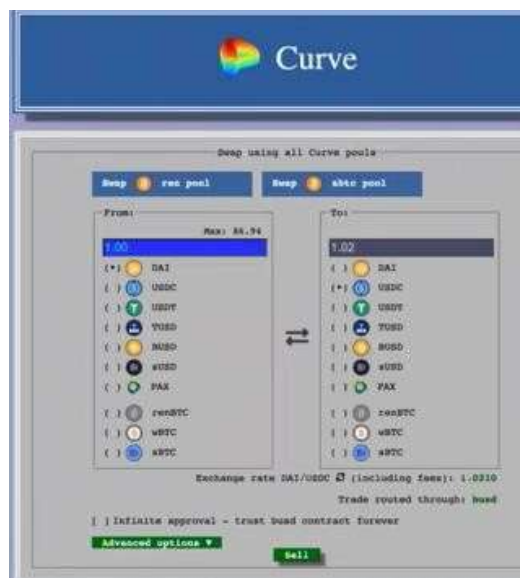
KEY TAKEAWAYS

- Slippage refers to all situations in which a market participant receives a different trade execution price than intended.
- Slippage occurs when the bid/ask spread changes between the time a market order is requested and the time an exchange or other market maker executes the order.
- Slippage occurs in all market venues, including equities, bonds, currencies, and futures.
- The final execution price vs. the intended execution price can be categorized as positive slippage, no slippage, or negative slippage.
- Slippage can be limited by not executing trades late in the day, investing in calm and liquid markets, and placing limit orders.

<https://www.investopedia.com/terms/s/slippage.asp>

curve finance is a decentralized exchange that really focused on swapping stable coins and made it a lot more efficient. this was super helpful for the purposes of slippage, one of the biggest problems with exchanges in general and especially decentralized exchanges was this idea of slippage that when I

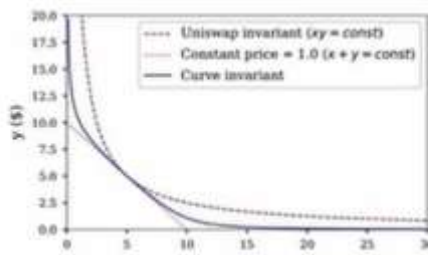
initiate a trade, the price at which my trade will execute will actually be a little bit different from what I initially clicked on because there's low liquidity and this would especially happen within larger orders. if I'm trying to sell like ten thousand each, my first few eth would be sold at a really good price and then my later eth will be sold at worse and worse prices because you're taking a bite out of the liquidity that exists on the protocol so you'd experience a lot of slippage on the price. the thing that curve finance did best is it tried to minimize the slippage that you experienced especially on stable coins and then between similarly priced assets.



you can see in the picture above like ren BTC or wBTC or BTC itself, these assets that are all supposed to have the same price to interact with different DeFi protocols, you need to kind of trade back and forth between them. what things like Uniswap would do is they would go from let's say you want to trade stable coin A to stable coin B, what they would do is they would trade stable coin A to eth because each was central for Uniswap and then trade each for stable coin B and that really resulted in a lot of slipping so you would see yourself putting in 500 and getting out like 488 dollars because of all that

transaction fees and the slippage that happened in between and curve really did its best to minimize this and allow people to trade directly between stable coins. it also gave people a lot yield and it was one of those money logos that you plugged in with other things like Yearn.Finance, people would have transactions on Yearn, take out what they got from Yearn and put that on curve and then overall in the system create a yield generating machine that gave them as many returns as possible which was a super interesting phenomenon that happened in DeFi summer.

- DEX and liquidity aggregator to make **swapping stablecoins** more efficient
- Low slippage, minimal impermanent loss
- Trade **directly between stablecoins**
- Significant yield fluctuation, acts as a **"money lego"** for other parts of DeFi



moving forward from that we have YAM finance, this was a kind of one of the more unfortunate events that happened during the DeFi summer.

it blew up quick and then kind of lost all of its money because of probably under auditing or lack of auditing for the smart contract.

what happened was that people saw it as a new and shiny project, a way to make more money and then the price skyrocketed or the total value locks skyrocketed to about 63 million and then back down to zero within the course of two days and this was really described as the perfect pump and dump setup and people really lost a lot of money on it. what happened is really a change in people's sentiments, YAM

finance was really one of the catalysts towards the cooldown of DeFi summer, people starting to understand that there's a high emphasis on smart contract auditing.

As we said before in the beginning of Yearn.Finance it gained a lot of traction and got hundreds of millions of dollars of value locked and people realized a couple days that based on this mark, the way that the smart contract, Andre, had access to the value locked in Yearn, he could have just taken the hundreds of millions of dollars that were there. to increase trust in his own product as a show of goodwill he burnt the like multi-sig keys that gave him access to that and really put a lot of people's fears to risk and YAM finance was an example of something that had a similar avenue for losing funds but that lacked the goodwill of the events involved in Yearn.Finance. on one hand it reduced a lot of trust in the DeFi space but also gave people something to focus on and kind of cemented in people's minds the importance of focusing on auditing.

- "Minimum valuable monetary experiment"
- **Market Value: 0 → \$63M → 0 in 2 Days**
- Massive demand for financial structure innovation, with nearly \$500M TVL captured
- Messari CEO: **"Perfect P&D Setup"**
- Unaudited source code led to rebase being able to mint an unlimited number of tokens

another of the events that had a good amount of effect on DeFi summer and DeFi space ship today is the Uniswap token drop. a lot of these projects that had governance tokens like compound and balancer, Uniswap in kind of a way to like stick to their guns of decentralization, they basically had this token drop where 400 Uni tokens were airdropped to all the 16 000 users or all the 16 000 wallets that I transacted on Uniswap in the past, this is kind of thought of as a DeFi stimulus and it was really their way of saying hey we are a decentralized platform and to show that we're a decentralized platform we want to

take our governance token and distribute it among everyone who has transacted on the system.

this is one of the most lucrative airdrops that has ever happened in the past. what happened at the time was as a result of events such as the YAM finance event where people lost trust in DeFi, many of the users really just took this as an excuse to dump on the market to take the funds that they got from this airdrop and then exit DeFi as soon as possible. this drop of the Uni token price happened right after the Uni drop.

- **Uniswap > Coinbase** in Monthly Volume
- 400 UNI Airdrop to 16,000 users from past:
Known as the **DeFi stimulus check**
- "Best token distribution of all time"
- Signaling **end of DeFi Summer** as UNI continues to plummet from former exodus

Moving on we have this idea of DeFi fall that basically happened post DeFi summer as a result of increase in DeFi scams among users.

during DeFi summer, DeFi really took center stage for crypto, people were talking about decentralized protocols more than anything else in crypto when things get gained a lot of traction and a lot of value fast especially when people get burned you'll usually see a cool down and that's exactly what happened with DeFi summer is that DeFi has uh seen a lot of changes and one of the things that contributed to the cooldown of DeFi was the spike of each gas prices, you found that people had to pay like tens sometimes hundreds of dollars in gas fees just to interact with these protocols. if you're playing with like a couple hundred or a thousand dollars and want to interact with these DeFi protocols it stops making a lot of sense when per transaction you have to pay that much amount of things that

much gas fees and finance smart chain showed up as a portrayed alternative to this Binance, the smart chain, had a certain number like a certain specified number of validators that had higher barriers of entry to B validator the number of validators was very limited so it was definitely a bit more centralized than the ethereum but as a result it was a lot faster in transactions and a lot lower gas fee so just provided an alternative avenue for people to interact with DeFi protocols.

- Increase in **DeFi scams** among influencers
- Supply of **Yield Farmers > Token Liquidity**
- DeFi Insured on Nexus Mutual has increased to \$100M: **more caution** with users
- **Cooling affected overall crypto market**
- Macro trends took steam away from market

DeFi is so much !!

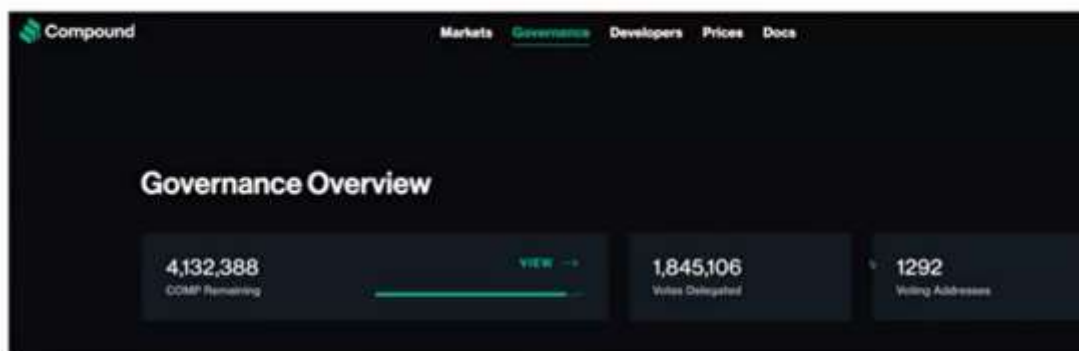
Let's highlight a little bit about what DeFi looks like as a universe right now and also what's next for DeFi. in the next diagram just how big DeFi is we mentioned maybe five or ten different projects but this list right here is just a subset of the people building around DeFi, trying different things in different sorts of categories the categories we've talked about like derivatives and trading and lending and stable claims and even more have popped up since this has come out.



Here's some more!

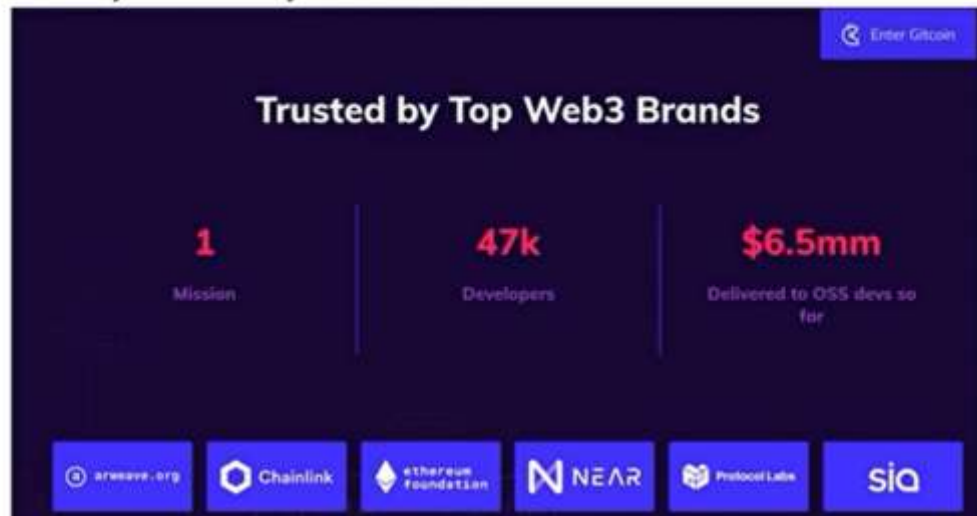


kind of highlighting the diversity in other ways we now also have diversity in the management of these DeFi protocols for something like compound where we talked about them doing decentralized governance we see the compound is now technically controlled and voted on by over 1200 voting addresses, it has grown significantly since then and now blockchain at berkeley is even part of that.



The picture below also shows diversity in development, kind of the way that we are developing these DeFi products.

Diversity... everywhere!



this is a screenshot from coin these guys are doing a bunch of cool grants for DeFi companies and DeFi developers we're seeing how many developers are around here indeed making it a really exciting place to be.

Future of DeFi

in the last few months we've had a lot happen in DeFi something you guys have probably caught on is the re-rise of NFTs and the hype cycle that we're in. there have been questions around what is it, what do NFTs mean for DeFi and there's a lot that they can mean, we have projects deploying on layer two. to scale layer two we'll talk about a little bit layer two and cross chain left and right we've talked about binance smart chain and the kind of overall movement to focusing on layer two and cross chain deployment, part of that cross-chain cycle was of course the binance smart chain hard fork which was pretty crazy going on, there were a few waves of these things called algorithmic stable coins which is highlighted by FEI protocols recent few weeks they are another one of these algorithmic coins that has gone through a lot in the last few weeks. the first step in Ethereum 2.0 release for further scaling on the Ethereum blockchain which is one of the five's biggest homes and much more has happened in the last few months

even though deep is past its manic summer the growth is hardly over.

- 🖼️ The re-rise of NFTs
- ⚡ Projects deploying on Layer 2 & Cross-Chain left and right
- 📈 Binance Smart Chain Hype Cycle
- 🏦 Waves of algorithmic stablecoins, highlighted by FEI
- 📝 The first step taken in Ethereum 2.0's release/migration!

And much much more!

DeFi may be past it's manic summer, but the growth is hardly over

aggregation all the way up we've kind of seen a couple layers of this but the overall question is how do we bring DeFi to the masses? and one answer is make it feel like they're barely touching a blockchain, they're barely touching things like Uniswap where you have to understand the risks of trading at a very deep level and you have to really understand almost how an AMM works we wanted to abstract things like that. for example we have things like DEX aggregators so now aggregators will look at different DEXes and give you a better price, the best price across these different indexes we have the smart investment managers so yEarn is a very big name and they basically allow you to deposit your crypto and they will smartly invest it in any which way.

The big question: how do we bring DeFi to the masses?

- One answer: make it feel like they're barely touching a blockchain!
 - DEX Aggregators (1inch, Matcha, dex.ag)
 - Smart Investment Managers (yEarn, Zapper, DefiSaver)

Find the **best prices** across
exchange networks

Try 'ETH / DAI' 🔍 Explore

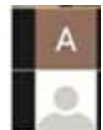
[1inch](#) [Balancer](#) [Uniswap](#) [Kyber](#) [Curve](#)

there are hopes that in the end it'll feel almost just like you are taking your USD or us dollars dropping it into some platform

but then in the end you will be using decentralized finance. privacy can be DeFi's core innovation and the problem is that it's not here yet right now in many of these public smart contract blockchains, everything is tied to an address and there are crazy stories of miners because they know and can see what you're trading front running you or trading right before you do to kind of manipulate prices and for them to take profit in for you to lose. it's a crazy type of thing and there's this really cool article if you just search up Ethereum dark forest that article will pop up and that's a great read um just to mention this.

but at the same time progress is being made here, things like shielded transactions and private swaps and things like that often powered by zero knowledge technology are really cool things to kind of google and learn about for how DeFi is looking to improve on the private aspect.

- Many believe that privacy can be DeFi's core innovation
 - But it's not here yet!
 - Everything is tied to an address
 - Scary mempool frontrunning (a great read 🧡)
- Progress is being made!
 - Dark pools, private swaps, often powered by zero knowledge proofs
 - All awesome things to google



Ethereum is a Dark Forest

By Dan Robinson and Georgios Konstantopoulos

 Dan Robinson · Aug 28 · 7 min read



This is a horror story.

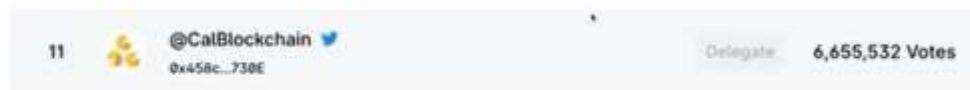
we've kind of talked about the decentralized execution of these financial products, when I'm using this financial product in DeFi I know that everything's going through this auditable smart contract but how about the way that updates are pushed out to a contract, the way that the entire community

or protocol evolves as a whole, things that are now being handed off from these protocols and teams themselves we should have to build them in the first place to now entire communities.

for example Aave was something like this who last fall literally handed over the keys which give them the ability to upgrade their smart contracts to the community so a vote has to happen for anything to change out there which is very brave and very bold. governance is definitely an experiment that blockchain Berkeley is lucky enough to be part of last December, we were delegated enough to be a top 10 boat holder on both compound and Uniswap so we as a regular voter, we vote on proposals, we are working to make proposals to continue to change and improve these DeFi protocols.

Decentralized Governance to the MAX

- We are pushing the standards for governance to a whole new level
 - Projects are literally handing over ownership of themselves to community! Aave for example
 - The rise of the "Fair Launch"
- Governance is a continued experiment with many open questions
 - How do you create a fair, engaged community?
 - Success stories balance out failure stories alike



any time of the last six months you couldn't enter an executive trade position without spending many tens of dollars at one time. how do you bring DeFi to the masses? you scale such that things like high fees don't have to exist so they're kind of two frontiers for scalability.

we talked about L1 scalability for something like Ethereum where DeFi is primarily, right now we have things like Ethereum 2.0 to look forward to which is coming. and then on another

side note an important note we've mentioned DeFi is not just Ethereum if DeFi exists on all sorts of other L1, other layer base layer chains.

we've heard about the binance chain other popular ones are solana mir polkadot is a really big one popping off for DeFi right now even companies in our startup accelerator were based on polka dot and they were doing DeFi.

Moving on different chains has also been another pattern which is very interesting and then on the L2 side of things especially on Ethereum, L2 has been the way that projects have turned to scale there. things like roll-ups are being used right now to be an L2 for many of these DeFi applications, things like Uniswap universe is moving to the optimism roll up or things like Aave working on side chains all sorts of other things like that . that is their short-term scalability solution because now you can trade on things like Uniswap on L2 for a fraction of a cent which is amazing.

Back to the first question: how do you bring DeFi to the masses??

...You **SCALE!**

- Two frontiers for scalability
 - L1
 - Ethereum 2.0 is coming! It's actually coming!
 - However, a **REALLY** important note -> DeFi != Ethereum!
 - Multi-chain DeFi!
 - L2
 - Rollups, Optimistic VMs, Sidechains, fancy!
 - Many projects have turned to L2 recently



Lecture 10

A blockchain powered future

<https://www.youtube.com/watch?v=c9HYeYFKRc&list=PLLkiRvMHnp6OIWkZVa85g2tv7NtlgoAID&index=10>

A blockchain powered future

We are gonna start off with a hypothetical future situation where your world is powered by blockchain. let's say in the morning you have just left an airbnb where you were vacationing and as you leave there are sensors that detect your departure, let's say it's a company called stockits and their smart contract confirms that during your time in the airbnb you didn't break anything and thus they pay you back your security deposit then you head over to Peet's coffee and you buy it from your smartphone and the coffee has an entire supply chain history like a fair trade certification and soil analysis from the farm that the coffee beans were from and that's the way that many products operate in this blockchain powered future, they have this option where you can understand where the ingredients originate from and can also include micro donations embedded into the price which helps to offset the price of carbon and water or contributes to charitable causes so you can see where the donations go for things like coffee beans at peet's coffee because they are operating through a public cryptocurrency, you call your car as you leave the coffee shop but it's not really yours it's actually plugged into an autonomous ad hoc swarm of vehicles readily available for use so it's communal transportation in a way accidents are rare in this blockchain-powered future but when they do happen the insurance agency responsible for the cars analyzes video footage through a machine learning trained bot and automatically gives the victim of the car accident, the appropriate payment which has been determined through an algorithm, however the insurance agency isn't exactly an insurance agency rather it's a smart contract that pulls together

the funds of a multitude of owners and charges people the exact expected value of car insurance without taking a profit it's almost like mutual aid. the near exact probability of accidents is known because every accident that has happened in this blockchain-powered future is recorded on the blockchain, you then head off to work where you have your occupation for the day however your work isn't really your work it's all voluntary because the blockchain ensures a universal basic income for everyone and you're an engineer and researcher by trade primarily because you want to help push the boundaries of human knowledge. your company isn't really a company either it's a decentralized autonomous organization consisting of individuals from and around the world who voluntarily signed up receive compensation and pay raises based off the quantity and quality of their research findings which are determined by a worldwide oracle network so that means that people can exit their contract at any time and your company itself receives revenue over time as measured by the net benefit to society your research brings so that's in direct opposition to having a social benefit versus a private benefit, this company is actually incentivized to operate in benefit of external sources rather than themselves they're motivated to behave selflessly and for the few functions of society left that still require a government it's run by a democratic feuder key, this is where markets decide which policy should be implemented and given a proposal to approve or reject. there are two prediction markets that are created and each contains one asset. one market corresponds to acceptance and the other to a rejection this completely removes the possibility of corrupt politicians who know little to nothing about the fields they write legislation for this eliminates a lot of bureaucracy nepotism and lobbying that occurs commonly in political landscapes. now on to the health

sector rather than an entirely blockchain run future it's more likely that the health care system in the united states will have a partial integration with the blockchain space and this is happening right now before our eyes for example anytime you go to the doctor you would be able to see your summary signed off by a doctor sent to you for a confirmation signature before it also goes to your insurance company you would have access to your electronic health records easily accessed by you when visiting other public hospitals and that is more transparent for you and only visible to you unless given authentication which would be possible by cryptography another example that's being worked on these days in the healthcare industry is putting the entire US pharmaceutical supply chain onto the blockchain the US FDA plans to have this implemented by 2023, so by the time some of you graduate keep your eyes out for this and in order to tackle the problem of counterfeit medicine which is a real problem, there's placebos in medical studies where people are voluntarily taking a sugar pill but when people are buying exorbitantly overpriced goods like insulin. it's really upsetting when they're being peddled false products thus this has actually been the case in some countries like Russia and other eastern European places. people are peddled false medication all the time and this is a way to help fix that.

Problems with blockchain

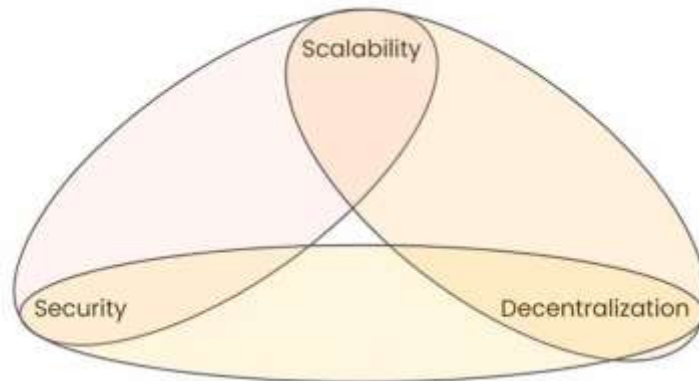
after taking you through this utopian society where everything is powered by blockchain and you kind of imagine this is what everyone in blockchain at berkeley thinks the future's gonna look like.

we're gonna show you why it's not happening why it's not super feasible right now and take you through a more realistic standpoint on blockchain. what is preventing adoption right

now, there are still a number of issues some of which we've mentioned in lectures, some of which you have brought up to us that are preventing widespread adoption of blockchain technology such as technical shortcomings so issues with the technology itself regulatory concerns which relates to governments and financial institutions and their regulations placed against blockchain technology and then we'll talk a little bit about maybe why they are doing that and then also usability issues which is kind of a function of how new and fresh the technology is and not necessarily there to stay but still preventing the scalability of blockchain technology. the biggest thing and this is something that you've definitely heard before, in conversations people love to bring this up is transaction throughput and specifically transactions per seconds or TPS. that's the biggest pitfall and really the biggest reason why people haven't been using bitcoin, Ethereum, all these other decentralized cryptocurrencies to do basic transactions because the technology at the moment can't really fit all those transactions, for example visa can support up to 65 000 transactions per second and they're supporting this around the world on average they only need about almost 2 000 transactions per second but if you run the numbers on bitcoin it can actually only support a maximum of 27 and you see why disparity in those two numbers and why that's a big problem.

	AVERAGE TPS	MAX TPS
BITCOIN	3.3	27
PAYPAL	193	450
VISA	1736.11	65000

moving forward we have the scalability trilemma which is another thing that we've talked about in lecture, but it's just detailing that there continue to be inherent trade-offs between these three properties, decentralization, scalability and security and decentralization.



we've talked a lot about that it's defined as a system being able to run in a scenario where each participant only has access to their resources and the entire system is basically run on each individual's resource so it's not in a centralized place.

scalability is what we were talking about, we want to scale this up to the masses for them to be able to use it.

Security, trying to secure those transactions and prevent against attacks. there's a trade-off between all these three things that's preventing financial systems from adopting blockchain technology at the moment. another technical problem is the oracle problem.

If your protocol uses off-chain data, how do you verify that it's accurate

You can't. You have to rely services called **oracles** that provide the data along with a proof of its validity.

However, you expose your protocol to the risk of a faulty oracle...

if we want to get really creative with how we use blockchain technology and build decentralized applications on it, something that we're probably going to want to use and we're going to see at some point is the need for external data to

come in and inform the protocol. so it's really nice to be able to have that data but at the same time that's coming in from off the chain but it's important that your application can verify that data is accurate and the problem with blockchain is that how can you actually verify that and how can you make that promise.

The simple like truth is that you can't, you have to rely on these services called oracles that can provide that data and prove that it's valid but when you do that you are exposing your protocol to the risk of a faulty oracle meaning that any time you're opening up your technology to outside data, you are opening up to the risk that someone on the outside could be lying to you and they can claim that it's true and create a valid proof for it when actually the data is invalid.

we're gonna bring up an example which is supply chain tracking.

Goal: Ensure that coffee was sourced ethically by recording a proof of ethical conduct for each link in the supply chain



Still requires someone/something (an oracle) to write in, "this was done ethically" or "this came from an ethical place."

when you're in that blockchain utopia and you go to buy your coffee, you are trusting that it's ethically sourced and you have a proof of ethical conduct for each link in the supply chain so you see where the coffee beans resource from, you see where it was brewed and on all that stuff but still at some point in that supply chain you require an external party to interact with the blockchain and sign off and say, at the very least scanning that coffee, it's QR code, yes this was ethically sourced and yes this came through here and yes I can confirm that I watched this from when it was still on the tree and now I'm sending it to you

but in reality that person could be lying and you never know if that human is doing some shady dealing and something under the table. whenever you're trusting the Oracle you're opening your protocol to that problem. shifting away from technical concerns with blockchain technology and more towards the regulatory side of things KYC and AML are these families of laws that govern financial institutions globally.

KYCs know your customer and AML is anti-money laundering laws, they essentially get the same thing. KYCs basically require that professionals make an effort to verify the identity and the risks associated with maintaining a business relationship.

if you want to sign up for a bank account, your the bank is going to require that you send them some sort of identity verification to make sure you're a real person and that you are who you say you are.

anti-monitoring money laundering laws are essentially what they sound like, they're risk-based programs to combat money laundering, so KYC goes hand-in-hand with AML because governments want to make sure that either knowingly or unknowingly banks aren't providing services for crime organizations, drug dealers or terrorist organizations they want to make sure that these banks understand their customers so that if anything illegal happens they're someone to blame. these laws inherently require that financial institutions re collect and maintain information about their users and their customers which that's totally against what blockchain technology stands for and all these decentralized protocols are built on pseudonymity, not knowing the information of the individuals involved and a lot of the regulatory concerns other than this, they stem in one way or another from pseudonymity and decentralization and how those tenants go against the traditional centralized banking institutions, they won't know

who to blame and that's a big problem in this area and similar regulatory issue but different as well.

Numerous regulatory concerns exist within blockchain, i.e. know your customer (KYC) and anti-money laundering (AML) laws.

These laws require financial institutions to collect certain personal information from their users, which goes against the core tenets of many decentralized protocols.

Many other regulatory concerns stem in one way or another from pseudonymity and decentralization; how do we know who to blame?

let's use a case example of libra, in 2019 facebook tried to start its own global stablecoin and it was centered in the fact that facebook had this wide-reaching arm into a lot of parts of the world that maybe couldn't trust their centralized financial institution for a stable coin, maybe they're experiencing hyperinflation but these people still had facebook so it was very well connected and facebook kind of took a step back and said we're well positioned to be a global payments network and they thought to provide a stable coin that's pegged to the dollar and you can use that securely no matter whether your local banks are fraught with corruption or hyperinflation or anything like that so a lot of people in the global economy took a look at that and said do we really want that to happen? because there was this chance that if facebook stable coin was really that much better than the centralized institutions in these countries then those citizens would then depend on facebook to make their monetary policy and everyone would succumb to the power of facebook and it would have looked pretty gnarly for the global economy. the European central bank, the French and German ministers of finance this group of seven nations, they came together to form this crypto regulation task force in response to

facebook's libra. facebook had to re-architect their protocol to support central bank issued stable coins representing different fiat currencies essentially saying that instead of having a single stable coin that was powered by facebook, that was gonna make up this global payment network, they had to integrate it with the central banks just to prevent widespread financial delinquency for all these governments that we're gonna be outsourced by facebook.

Facebook tried announced Libra, a global stablecoin, in June of 2019.

Met w/ concern for the global economy from:

- European Central Bank
- French & German ministers of finance
- G7 nations, which formed crypto regulation task force in response

Had to re-architect the protocol to support central bank-issued stablecoins representing different fiat currencies.

going into usability issues, the complicated user interface is just a factor of how early we are on in this technology. in the early stages user flow is not as polished, it's not gonna look like facebook it's not gonna look like spotify, it's gonna look like pure html and user has to know about those technicalities if you're going to be interacting with the blockchain itself you're going to have to understand how it works at a deep level but the idea is that with more mainstream integrations and as more people learn about the technology, the user experience is going to improve and it's already improved a lot.

Complicated UI/UX

- In the early stages, user flow is not as polished!
- Currently, the end-user has to know about the technicalities
- With more mainstream integrations, UI/UX is improving



usability relating to the preceding reputation, you may have heard of silk road, it was this kind of online black market, anonymous marketplace, and people would pay for these illegal goods and bitcoin because it couldn't be tracked and the guy who started is now arrested, FBI caught him, it was a problem because people in the early days associated bitcoin, with these types of websites because these websites use bitcoin towards their list of activity and it's not the technology itself that's corrupt at all it's you create a technology and people are gonna do what people do and they're going to use it for whatever they want. in the case of silk road they were using bitcoin because it was pseudonymous.

still a lot of people when you say bitcoin they're like wasn't that used in the black market. if you look at the wikipedia page for silk road there's a part that says the FBI initially seized 26 000 bitcoins from accounts on silk road worth approximately 3.6 million at the time and this was in early october of 2013 and that's worth 1.5 billion dollars today and then later in october the FBI reported that it seized an additional 144 000 bitcoins worth 28.5 million at the time and today that's worth 7.9 billion dollars.

And again we have this huge social media hack that happened last summer, they hacked into apple Elon musk, Kim Kardashian and on they tweeted we are giving back to our community, we support bitcoin and we believe you should too, all bitcoin central address below will be sent back to you double and a lot of people believe this. this address received hundreds of thousands of dollars worth of bitcoin at the time which is hundreds of millions at this point but again that also contributed to the reputation of these hackers are using bitcoin.

- In the early stages, user flow is not as polished!
- Currently, the end-user has to know about the technicalities
- With more mainstream integrations, UI/UX is improving



going over all those problems with blockchain we can see it's a rocky transition to go from traditional institutions into those powered by blockchain technology the more prominent blockchain becomes through bitcoin Ethereum DeFi the less power traditional finance systems have. now we can take a step back and say okay maybe this is why they're fighting it so much because the goal of cryptocurrencies and not all cryptocurrencies but bitcoin for example is to replace financial and government systems then wouldn't it make sense that those existing systems would resist that change and use the tools in their power such as regulations and laws to prevent that change from occurring because they're out of business if this becomes mainstream. instead of jumping straight into that utopian blockchain-powered future we need to find a way to cooperate and co-exist first which is kind of like the world we're living in today and then make that leap.

The more prominent blockchain becomes, through Bitcoin, Ethereum, Decentralized Finance, the less power traditional finance systems have.

If the goal is to replace financial and government systems, those existing systems will resist the change or bring it about slowly through regulations and laws.

May have to find a way to cooperate and coexist first, then make the leap.

a transition nonetheless, we are still transitioning to that blockchain technology powered future, coinbase stock debuted on Nasdaq (The Nasdaq Stock Market is an American stock

exchange based in New York City. It is the most active stock trading venue in the US by volume, and ranked second on the list of stock exchanges by market capitalization of shares traded, behind the New York Stock Exchange) last week in a direct listing as the premier cryptocurrency company go public which was super exciting. it shows that we're one step closer towards public adoption of this technology. China has a new statebacked digital currency which is not decentralized so it's a little different but it's still moving towards digital currencies, a couple in San Francisco exchanged digital NFTs as wedding rings

Some current event headline:

- Coinbase Stock Debuts On Nasdaq In Direct Listing As The Premier Cryptocurrency Company To Go Public
- China Leads The World With New State-Backed Digital Currency
-

The blockchain space is still VERY young, perfect time to join the journey!

the space is still very young but it's the perfect time to join the journey.

Get involved

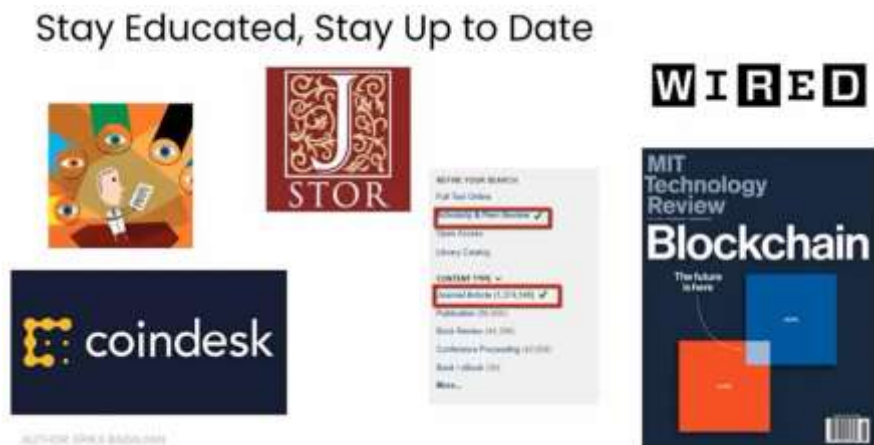
we'll take you through some of the ways that you can get involved. now that you know the pros and cons of blockchain we recommend that you get started with your educational journey even more than that. We recommend that you check out the educational resources that start at uc berkeley so that would be blockchain at Berkeley and that would be all our affiliate organizations.

Start with what B@B has to offer

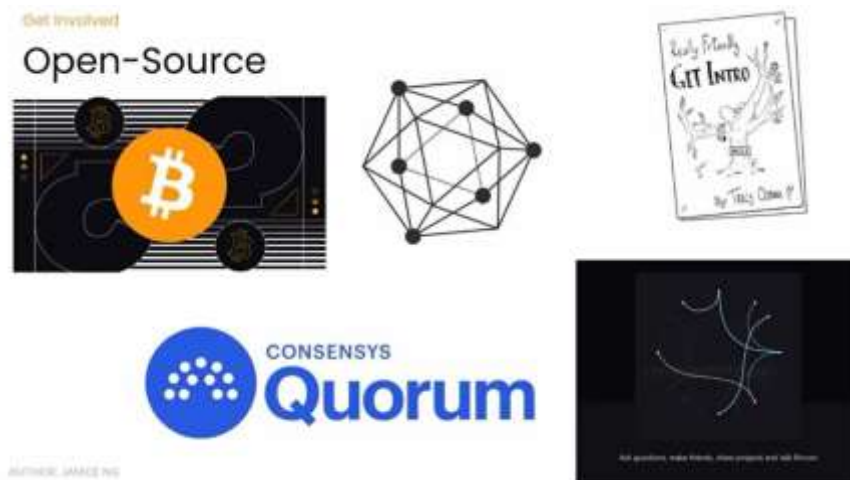


we have mentioned the different decals that we offer the different educational courses that we're ramping up so we have an EDX course for both blockchain technology as well as bitcoin and cryptocurrencies and you can get a certificate through that they have been taught to over a hundred and sixty thousand students worldwide and we've had so many reviews from students saying that they sincerely appreciate the content we've provided and that we've allowed them to enter the space because it can be super hard to get started on your blockchain journey without having someone to teach it to you because there's a lot out there, we also have our decal which you've taken now but we also have a blockchain for developers decal for when you want to get into the nitty-gritty of coding and maybe making a project on that has to do with being on the blockchain or involving smart contracts then we also have the blockchain at berkeley accelerator it's a joint program between the charges center for entrepreneurship and technology. We recommend that you all check out the public events that the accelerator puts on it's a great way to get acquainted with what the blockchain startup world looks like and it sort of gives you an upper hand on what the industry is pivoting towards. it's always good to get that information in there. another recommendation we have for you is people on the internet lie

sometimes and that happens especially in academia and in topics as trend heavy and buzzword heavy as blockchain so it's really important to respect the sources and be picky with the sources that you choose when reading about blockchain topics so some suggestions we have are websites like MIT technology review wired magazine jstor.



you also need to refine your search for these research papers that talk about blockchain technology or emerging topics within the field, you want to make sure that your topics are scholarly and peer-reviewed and you also want to make sure that it's occurred in a journal article which means that it had to be approved through a process and you also want to maybe check out coindesk and verify who the authors are of your paper for example if you're unsure about the carbon consumption of bitcoin mining one author could be telling you one thing and another could be telling you another but it's important to know where their interest lies as you go on your educational journey through the field otherwise you'll be believing everything people say.



we would also recommend that you start building on open source platforms that are available, check out what bitcoin core has to offer, check out what quorum which is a project part of consensus has to offer and then we also recommend this git intro it's a little zine in a very friendly way approaching how to do get polls and understand how github works from someone who's decidedly untechnical.

Observe the Blockchain Ecosystem



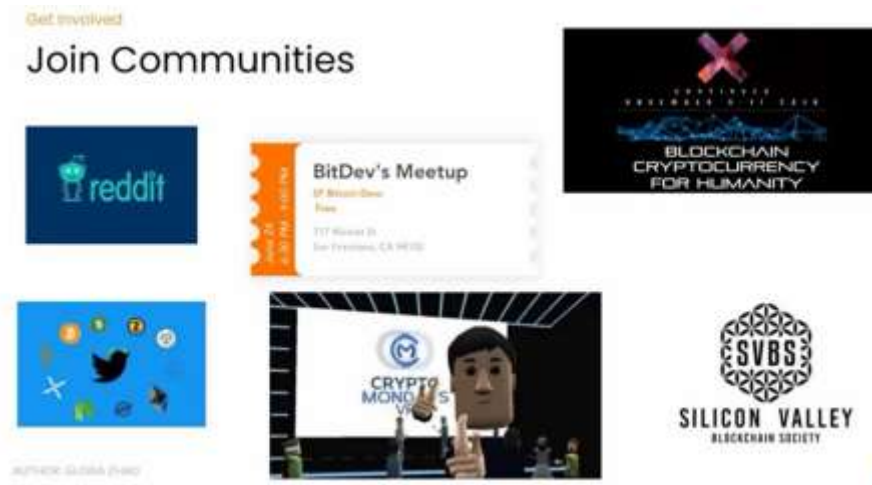
we also want you to recom on run we want to recommend that you observe the blockchain ecosystem you can do that by um subscribing to apple news and saying that you're interested in the cryptocurrency for articles that are flagged as that and when you're downloading investing apps like coinbase or robinhood making sure to check the news that they have on the bottom panel of the app making sure that you're always aware of who's emerging from the space, who's leaving the space for example DeFi was not a thing a couple of years ago but it's

really emerged and exploded and that may be something similar to DeFi will happen right in front of your eyes and it'll be cool to have the upper hand on that knowledge. there's always companies coming and going and we really recommend that you keep your finger on the pulse of the blockchain ecosystem.

Blockchain Careers Aren't Exclusively Technical



We also want to use this as a reminder that if you're interested in the blockchain space career-wise but technical aspects or computer science coding isn't something you envision yourself doing that's okay blockchain careers aren't exclusively just sitting in front of a computer and coding there's different ways that you could exercise your mind you could be a designer, you could be a technical writer and help to demystify blockchain concepts for people, you could be a researcher working with, from an interdisciplinary perspective on fields like psychology or anthropology or English language and as long as you have that interest internally for the blockchain space it doesn't matter if you might not think that your technical skills are strong enough that's also probably just you doubting yourself.



other recommendation is to join blockchain communities, there's no better place than the bay area for that, there's especially during this internet era and especially during covid so much has sprung up from our screens and there's so many ways to get involved, you could find your blockchain community on reddit or twitter or discord find a group of people who are interested in like-minded investing or investigating you could start a white paper circle with your friends where you talk about research topics that interest you in the space blockchain at berkeley also holds meetups pretty often. it's also important for you guys to know that blockchain at berkeley which is right here holds opportunities to mingle with people in the industry all the time.

Continue your Education



we also hope that after you exhaust all the opportunities that uc berkeley and blockchain berkeley have to offer that that doesn't stop your blockchain education journey so we'd highlight a couple courses for those of you who wouldn't be

satisfied with just what we have to offer, there's some really awesome stuff going on at the harvard law school, there's a great class going on at duke university on topics that are a little more niche focused than what blockchain berkeley offers so for example if you're really interested in law school and you want to understand how blockchain smart contract law operates, check out the harvard law school class or if you're really interested in blockchain from an entrepreneurial or managerial perspective check out that duke course i also want to highlight a chain code course that's occurring relatively soon and we would really recommend that you spend a weekend going through the bitcoin core book and really understand um the nuances of blockchain even pull out the blockchain for developers curriculum and then just go full speed ahead into that chain code class and really take it by storm.