Introduction to Cryptocurrency

Fall 2025

Joseph Hall

Georgia Institute of Technology

November 4, 2025

Roadmap

Roadmap

Hash Functions

- A **hash function** takes some input string, and "chaotically" spits out an output string. Demo
- Servers (to which you log in) store hashes, not passwords!

Hash Functions

- A **hash function** takes some input string, and "chaotically" spits out an output string. Demo
- Servers (to which you log in) store hashes, not passwords!
- "Chaotic" means deterministic but unpredictable: hash of a number never changes, but knowing hash of N doesn't reveal anything about hash of $N + 1$.
  - **One-way:** Can calculate outcomes, but can't back out initial conditions
  - **Collision Resistance:** Lots of outcomes are possible, so two initial conditions rarely result in same outcome (a "collision")

Hash Functions

A cryptographic hash function is a mapping $H : M \to D$ from a set M of possible messages of any length to a set D of digests, for example the set of strings of 0s and 1s of a fixed length, such as 256. See Ramzan (2012).

[1] digital signature schemes do preserve some properties of x as we will see

Hash Functions

A cryptographic hash function is a mapping $H : M \to D$ from a set M of possible messages of any length to a set D of digests, for example the set of strings of 0s and 1s of a fixed length, such as 256. See Ramzan (2012).

For usefulness, a cryptographic hash function H should be easily computable and, among other properties, be:

1. **Collision resistant**, meaning that H is "nearly injective." That is, it should be extremely rare to have $H(x) = H(y)$ unless $x = y$.

[1] digital signature schemes do preserve some properties of x as we will see

Hash Functions

A cryptographic hash function is a mapping $H : M \to D$ from a set M of possible messages of any length to a set D of digests, for example the set of strings of 0s and 1s of a fixed length, such as 256. See Ramzan (2012).

For usefulness, a cryptographic hash function H should be easily computable and, among other properties, be:

1. **Collision resistant**, meaning that H is "nearly injective." That is, it should be extremely rare to have $H(x) = H(y)$ unless $x = y$.
2. **Difficult to invert**. That is, it should be extremely hard to guess x, or properties of x[1], from $H(x)$.

[1] digital signature schemes do preserve some properties of x as we will see

Hash Functions

A cryptographic hash function is a mapping $H : M \to D$ from a set M of possible messages of any length to a set D of digests, for example the set of strings of 0s and 1s of a fixed length, such as 256. See Ramzan (2012).

For usefulness, a cryptographic hash function H should be easily computable and, among other properties, be:

1. **Collision resistant**, meaning that H is "nearly injective." That is, it should be extremely rare to have $H(x) = H(y)$ unless $x = y$.
2. **Difficult to invert**. That is, it should be extremely hard to guess x, or properties of x[1], from $H(x)$.
3. **Extremely sensitive**. Even small changes in the message should usually produce large changes in the digest.

[1] digital signature schemes do preserve some properties of x as we will see

Hash Functions

A cryptographic hash function is a mapping $H : M \to D$ from a set M of possible messages of any length to a set D of digests, for example the set of strings of 0s and 1s of a fixed length, such as 256. See Ramzan (2012).

For usefulness, a cryptographic hash function H should be easily computable and, among other properties, be:

1. **Collision resistant**, meaning that H is "nearly injective." That is, it should be extremely rare to have $H(x) = H(y)$ unless $x = y$.
2. **Difficult to invert**. That is, it should be extremely hard to guess x, or properties of x[1], from $H(x)$.
3. **Extremely sensitive**. Even small changes in the message should usually produce large changes in the digest.

In **digital signature schemes** (as we will see), a "digest" is a **signed message**.

[1] digital signature schemes do preserve some properties of x as we will see

Collision Resistance and Invertability

Question: Suppose a function H maps the integers 1, 2, 3, 4, 5, 6 to the integers 0, 1, 2, 3, 4, 5 by letting H(x) = x - 1. For example, H(5) = 4. Is H collision resistant?

Collision Resistance and Invertability

Question: Suppose a function H maps the integers 1, 2, 3, 4, 5, 6 to the integers 0, 1, 2, 3, 4, 5 by letting H(x) = x - 1. For example, H(5) = 4. Is H collision resistant?

Answer: Yes, it is. Observe that $H(x) - H(y) = x - y$ in this case, so if $x \neq y$, then $H(x) \neq H(y)$

Collision Resistance and Invertability

Question: Suppose a function H maps the integers 1, 2, 3, 4, 5, 6 to the integers 0, 1, 2, 3, 4, 5 by letting H(x) = x - 1. For example, H(5) = 4. Is H collision resistant?

Answer: Yes, it is. Observe that $H(x) - H(y) = x - y$ in this case, so if $x \neq y$, then $H(x) \neq H(y)$

Question: is $H(.)$ difficult to invert?

Collision Resistance and Invertability

Question: Suppose a function H maps the integers 1, 2, 3, 4, 5, 6 to the integers 0, 1, 2, 3, 4, 5 by letting H(x) = x - 1. For example, H(5) = 4. Is H collision resistant?

Answer: Yes, it is. Observe that $H(x) - H(y) = x - y$ in this case, so if $x \neq y$, then $H(x) \neq H(y)$

Question: is $H(.)$ difficult to invert?

Answer: No, it is easy to invert. In this case $H^{-1}(x) = x + 1$

Collision Resistance and Invertability

Question: Suppose a function H maps the integers 1, 2, 3, 4, 5, 6 to the integers 0, 1, 2, 3, 4, 5 by letting H(x) = x - 1. For example, H(5) = 4. Is H collision resistant?

Answer: Yes, it is. Observe that $H(x) - H(y) = x - y$ in this case, so if $x \neq y$, then $H(x) \neq H(y)$

Question: is $H(.)$ difficult to invert?

Answer: No, it is easy to invert. In this case $H^{-1}(x) = x + 1$

Question: is $H(.)$ a useful cryptographic **hash function**?

Collision Resistance and Invertability

Question: Suppose a function H maps the integers 1, 2, 3, 4, 5, 6 to the integers 0, 1, 2, 3, 4, 5 by letting H(x) = x - 1. For example, H(5) = 4. Is H collision resistant?

Answer: Yes, it is. Observe that $H(x) - H(y) = x - y$ in this case, so if $x \neq y$, then $H(x) \neq H(y)$

Question: is $H(.)$ difficult to invert?

Answer: No, it is easy to invert. In this case $H^{-1}(x) = x + 1$

Question: is $H(.)$ a useful cryptographic **hash function**?

Answer: No, hash functions must be hard to invert.

Hash Functions

Hash functions are the basis for **Proof of Work**

- By publishing $y = H(x)$, anyone can check that they know the real x without x ever becoming public information

Hash Functions

Hash functions are the basis for **Proof of Work**

- By publishing $y = H(x)$, anyone can check that they know the real x without x ever becoming public information
  - Easy to compute $H(x)$ given $H(\cdot)$ and x

Hash Functions

Hash functions are the basis for **Proof of Work**

- By publishing $y = H(x)$, anyone can check that they know the real x without x ever becoming public information
  - Easy to compute $H(x)$ given $H(\cdot)$ and x
  - If someone changes x to $\tilde{x}$, even by one letter, $H(\tilde{x})$ will be very different!
  - Used in checksums for file downloading

Hash Functions

Hash functions are the basis for **Proof of Work**

- By publishing $y = H(x)$, anyone can check that they know the real x without x ever becoming public information
  - Easy to compute $H(x)$ given $H(\cdot)$ and x
  - If someone changes x to $\tilde{x}$, even by one letter, $H(\tilde{x})$ will be very different!
  - Used in checksums for file downloading
- Application: **Mining and proof-of-work**
  - No way to discover x given $H(x)$ beside **brute force**!

Hash Functions

Hash functions are the basis for **Proof of Work**

- By publishing $y = H(x)$, anyone can check that they know the real x without x ever becoming public information
  - Easy to compute $H(x)$ given $H(\cdot)$ and x
  - If someone changes x to $\tilde{x}$, even by one letter, $H(\tilde{x})$ will be very different!
  - Used in checksums for file downloading
- Application: **Mining and proof-of-work**
  - No way to discover x given $H(x)$ beside **brute force**!
  - The Secure Hash Algorithm 1 ("**SHA1**") takes between 2^{63} - 2^{80} evaluations to find a collision (x, y such that $H(x) = H(y)$) (depending on how clever you are)

Hash Functions

Hash functions are the basis for **Proof of Work**

- By publishing $y = H(x)$, anyone can check that they know the real x without x ever becoming public information
  - Easy to compute $H(x)$ given $H(\cdot)$ and x
  - If someone changes x to $\tilde{x}$, even by one letter, $H(\tilde{x})$ will be very different!
  - Used in checksums for file downloading
- Application: **Mining and proof-of-work**
  - No way to discover x given $H(x)$ beside **brute force**!
  - The Secure Hash Algorithm 1 ("**SHA1**") takes between 2^{63} - 2^{80} evaluations to find a collision (x, y such that $H(x) = H(y)$) (depending on how clever you are)
  - Bitcoins are mined by brute-forcing hashes to find xs that produce $H(x)$ with leading zeroes.

Digital Signature Schemes

Requirements:

- JPH has a public key everyone knows
- JPH has a private key only JPH knows
  - **Non-invertability** means the public key doesn't reveal the private key

Digital Signature Schemes

Requirements:

- JPH has a public key everyone knows
- JPH has a private key only JPH knows
  - **Non-invertability** means the public key doesn't reveal the private key
- JPH can "sign" messages: use the private key to alter the message in a way that proves JPH created his public key
- Public key allows **verification**, but not **signature production**

Digital Signature Schemes

Requirements:

- JPH has a public key everyone knows
- JPH has a private key only JPH knows
  - **Non-invertability** means the public key doesn't reveal the private key
- JPH can "sign" messages: use the private key to alter the message in a way that proves JPH created his public key
- Public key allows **verification**, but not **signature production**
  - Using public key, anyone can verify signature correctness: JPH used his private key to sign the message

Digital Signature Schemes

Requirements:

- JPH has a public key everyone knows
- JPH has a private key only JPH knows
  - **Non-invertability** means the public key doesn't reveal the private key
- JPH can "sign" messages: use the private key to alter the message in a way that proves JPH created his public key
- Public key allows **verification**, but not **signature production**
  - Using public key, anyone can verify signature correctness: JPH used his private key to sign the message
  - But, using the public key, can't generate new valid signatures

Digital Signatures, Formally

A **Digital Signature Scheme** is a combination of Three Functions:

- A function to generate **Public Keys**: $PrK \Rightarrow PuK$

Digital Signatures, Formally

A **Digital Signature Scheme** is a combination of Three Functions:

- A function to generate **Public Keys**: $PrK \Rightarrow PuK$
- A function to generate **Signed Messages**: $(PrK, M) \Rightarrow M'$

Digital Signatures, Formally

A **Digital Signature Scheme** is a combination of Three Functions:

- A function to generate **Public Keys**: $PrK \Rightarrow PuK$
- A function to generate **Signed Messages**: $(PrK, M) \Rightarrow M'$
- A function to **Verify** signed messages: $(PuK, M, M') \Rightarrow \{0, 1\}$

Digital Signatures, Formally

A **Digital Signature Scheme** is a combination of Three Functions:

- A function to generate **Public Keys**: $PrK \Rightarrow PuK$
- A function to generate **Signed Messages**: $(PrK, M) \Rightarrow M'$
- A function to **Verify** signed messages: $(PuK, M, M') \Rightarrow \{0, 1\}$

These functions ($\Rightarrow$) are one-way, chaotic functions!

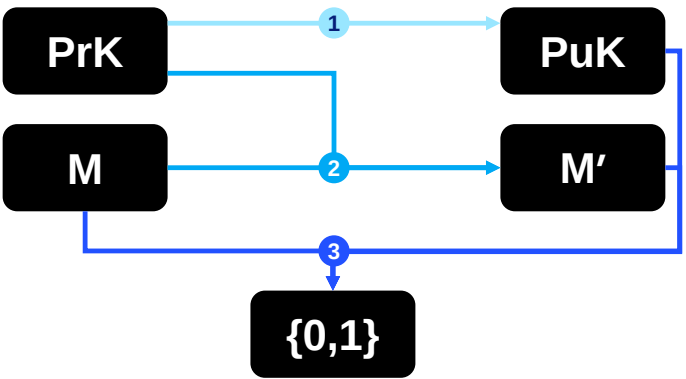
Digital Signatures, Formally

A **Digital Signature Scheme** is a combination of Three Functions:

- A function to generate **Public Keys**: $PrK \Rightarrow PuK$
- A function to generate **Signed Messages**: $(PrK, M) \Rightarrow M'$
- A function to **Verify** signed messages: $(PuK, M, M') \Rightarrow \{0, 1\}$

These functions ($\Rightarrow$) are one-way, chaotic functions!
In Bitcoin, these "messages" encode a list of transfers of blocks from one owner to another!

Digital Signature Schemes, Visually



Ok, but how do these signatures actually work?

ECE 6208 teaches this.

An easy scheme to implement from scratch is called ElGamal.

Ok, but how do these signatures actually work?

ECE 6208 teaches this.

An easy scheme to implement from scratch is called ElGamal. ("The Beauty" in Arabic)

Uses of Merkle Trees

A Merkle Tree is a way of efficiently storing the required information to ensure that a set of documents or messages have not been tampered with or corrupted.

Uses of Merkle Trees

A Merkle Tree is a way of efficiently storing the required information to ensure that a set of documents or messages have not been tampered with or corrupted.

Merkle Trees are used in many applications to solve problems such as:

- Maintaining integrity of files stored on Dropbox.com, or *file outsourcing*
- Proving a transaction between Alice and Bob occurred, or *set membership*
- Transmitting files over unreliable channels, or *anti-entropy*

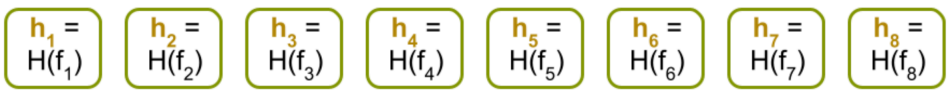
Construciton of Merkle Trees

Start with, say, 8 files $(f_1 ... f_8)$, and a hash function $H(.)$

Construciton of Merkle Trees

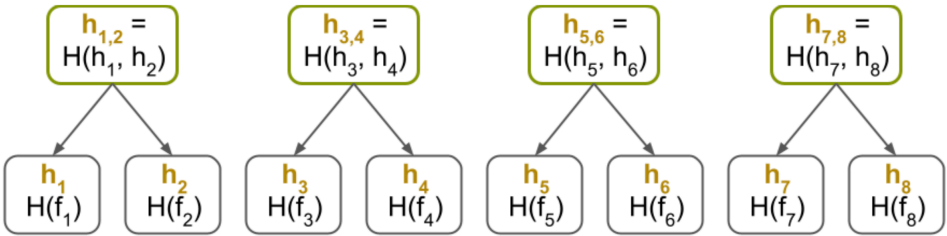
Start with, say, 8 files $(f_1 ... f_8)$, and a hash function $H(.)$

Start by hashing each file $H(f_i)$:



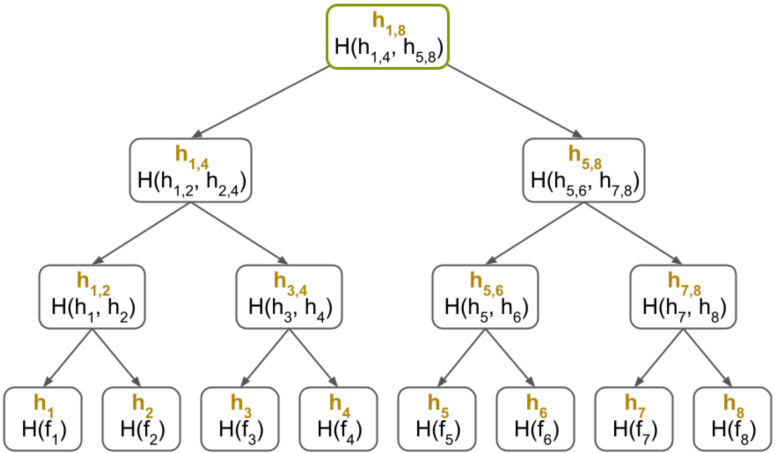
Construction of Merkle Trees

Then, create the second layer of the tree by hashing adjacent nodes together (this can be as simple as concatenating them!)



Construction of Merkle Trees

Continue until you have created an entire "tree":



Construction of Merkle Trees

Question: Why is this tree upside-down?

Construction of Merkle Trees

Question: Why is this tree upside-down?
Answer: Because Computer Scientists have never gone outside, a necessary prerequisite to see a real tree

Verification with Merkle Trees

Upload your files $(f_1 ... f_8)$ **and** your Merkle Tree to Dropbox, storing only the **Merkle Root** $h_{1,8}$ locally.

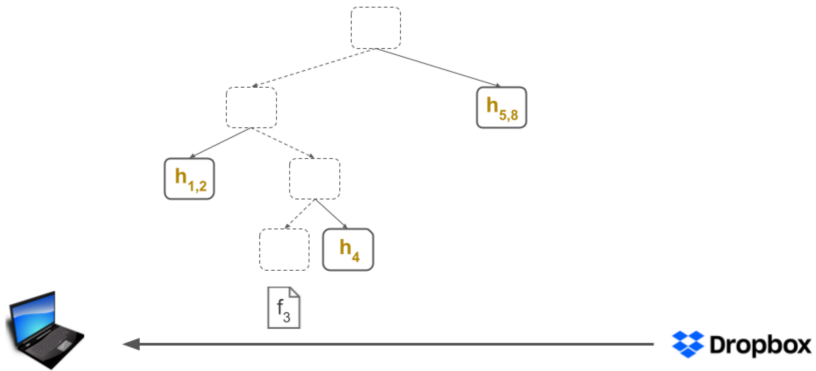
Verification with Merkle Trees

Upload your files $(f_1 ... f_8)$ **and** your Merkle Tree to Dropbox, storing only the **Merkle Root** $h_{1,8}$ locally.

Later, ask for the file you want, say f_3, but make sure to also ask for the **Merkle Proof**: the hashes needed to compute $h_{1,8}$ from the desired **Merkle Leaf** (f_3)

Verification with Merkle Trees

The **Merkle Proof** lets you verify that you downloaded the real, original f_3:



Advantages of Merkle Trees

Question: Why not just store the hash of every file? Wouldn't that be simpler?

Advantages of Merkle Trees

Question: Why not just store the hash of every file? Wouldn't that be simpler?
Answer: That would require storing 1 hash per document, so your local storage would scale linearly with number of files. The Merkle Tree requires storing **only** the Merkle Root, a single hash, **for any number of files**!

Advantages of Merkle Trees

Question: Why not just store the hash of every file? Wouldn't that be simpler?
Answer: That would require storing 1 hash per document, so your local storage would scale linearly with number of files. The Merkle Tree requires storing **only** the Merkle Root, a single hash, **for any number of files**!

Question: The server stores (h_4, $h_{1,2}$, and $h_{5,8}$) for you. Can't the server modify these to make the Merkle Proof appear true, even if f_3 is changed to $\tilde{f}_3$?

Advantages of Merkle Trees

Question: Why not just store the hash of every file? Wouldn't that be simpler?
Answer: That would require storing 1 hash per document, so your local storage would scale linearly with number of files. The Merkle Tree requires storing **only** the Merkle Root, a single hash, **for any number of files**!

Question: The server stores (h_4, $h_{1,2}$, and $h_{5,8}$) for you. Can't the server modify these to make the Merkle Proof appear true, even if f_3 is changed to $\tilde{f}_3$?
Answer: The cryptographic property of **Non-Invertability** ensures that it is prohibitively difficult to create a false Merkle proof.

Advantages of Merkle Trees

Question: Why not just store the hash of every file? Wouldn't that be simpler?
Answer: That would require storing 1 hash per document, so your local storage would scale linearly with number of files. The Merkle Tree requires storing **only** the Merkle Root, a single hash, **for any number of files**!

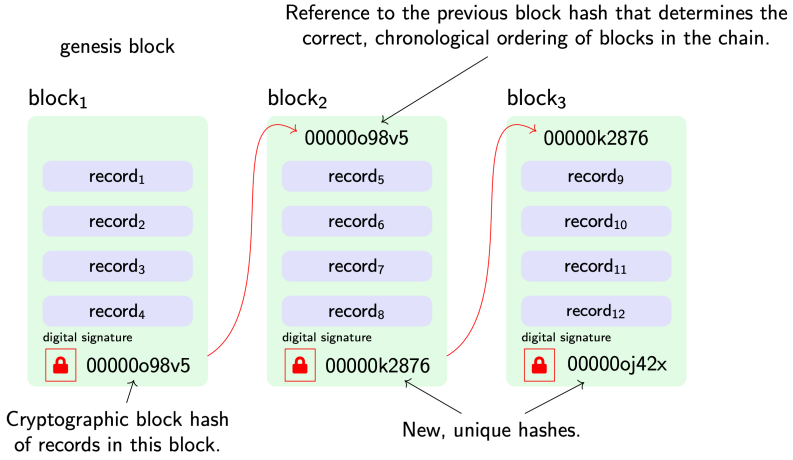
Question: The server stores (h_4, $h_{1,2}$, and $h_{5,8}$) for you. Can't the server modify these to make the Merkle Proof appear true, even if f_3 is changed to $\tilde{f}_3$?
Answer: The cryptographic property of **Non-Invertability** ensures that it is prohibitively difficult to create a false Merkle proof.

For other (analogous) uses of Merkle Trees including Transaction Verification and Transmitting files robustly over error-prone communication channels, see Tomescu (2020)

Roadmap

What is a Blockchain?

A **Blockchain** is a list of records that are linked together by hash functions



A Centralized Blockchain

Say that I, JPH, own and operate a private server which records a Blockchain. I don't know the users' private keys, but I control everything that is written on the blockchain. Can I:

A Centralized Blockchain

Say that I, JPH, own and operate a private server which records a Blockchain. I don't know the users' private keys, but I control everything that is written on the blockchain. Can I:

- Create false messages?

A Centralized Blockchain

Say that I, JPH, own and operate a private server which records a Blockchain. I don't know the users' private keys, but I control everything that is written on the blockchain. Can I:

- Create false messages?
  - No: I don't have the private keys to create messages that can be verified as true.

A Centralized Blockchain

Say that I, JPH, own and operate a private server which records a Blockchain. I don't know the users' private keys, but I control everything that is written on the blockchain. Can I:

- Create false messages?
  - No: I don't have the private keys to create messages that can be verified as true.
- Cancel or revert messages?

A Centralized Blockchain

Say that I, JPH, own and operate a private server which records a Blockchain. I don't know the users' private keys, but I control everything that is written on the blockchain. Can I:

- Create false messages?
  - No: I don't have the private keys to create messages that can be verified as true.
- Cancel or revert messages?
  - Yes: I can refuse to include a new block or roll back the blockchain to a previous state.

A Centralized Blockchain

Say that I, JPH, own and operate a private server which records a Blockchain. I don't know the users' private keys, but I control everything that is written on the blockchain. Can I:

- Create false messages?
  - No: I don't have the private keys to create messages that can be verified as true.
- Cancel or revert messages?
  - Yes: I can refuse to include a new block or roll back the blockchain to a previous state.
- Selectively cancel three messages ago, but not the last two?

A Centralized Blockchain

Say that I, JPH, own and operate a private server which records a Blockchain. I don't know the users' private keys, but I control everything that is written on the blockchain. Can I:

- Create false messages?
  - No: I don't have the private keys to create messages that can be verified as true.
- Cancel or revert messages?
  - Yes: I can refuse to include a new block or roll back the blockchain to a previous state.
- Selectively cancel three messages ago, but not the last two?
  - No: the hashes of the last two messages would be wrong if the middle hash is missing.

A Centralized Blockchain

Centralized Blockchains are in fact widely-used:

- Binance Smart Chain
- Coinbase Base
- Ripple XRP
- "Permissioned" or "Private" Blockchains use **Proof of Authority**: the admin's public key *per se* is hard-coded into the protocol.

A Centralized Blockchain

Centralized Blockchains are in fact widely-used:

- Binance Smart Chain
- Coinbase Base
- Ripple XRP
- "Permissioned" or "Private" Blockchains use **Proof of Authority**: the admin's public key *per se* is hard-coded into the protocol.

When are these better than just a spreadsheet?

A Centralized Blockchain

Centralized Blockchains are in fact widely-used:

- Binance Smart Chain
- Coinbase Base
- Ripple XRP
- "Permissioned" or "Private" Blockchains use **Proof of Authority**: the admin's public key *per se* is hard-coded into the protocol.

When are these better than just a spreadsheet?

- Often they aren't: Chase, VISA, Zelle, and Federal Reserves are all just spreadsheets at the end of the day

A Centralized Blockchain

Centralized Blockchains are in fact widely-used:

- Binance Smart Chain
- Coinbase Base
- Ripple XRP
- "Permissioned" or "Private" Blockchains use **Proof of Authority**: the admin's public key *per se* is hard-coded into the protocol.

When are these better than just a spreadsheet?

- Often they aren't: Chase, VISA, Zelle, and Federal Reserves are all just spreadsheets at the end of the day
- Centralized Blockchains add one extra level of security: create a transaction history that can't be arbitrarily tampered with: only censored or rewound.

A Centralized Blockchain

Centralized Blockchains are in fact widely-used:

- Binance Smart Chain
- Coinbase Base
- Ripple XRP
- "Permissioned" or "Private" Blockchains use **Proof of Authority**: the admin's public key *per se* is hard-coded into the protocol.

When are these better than just a spreadsheet?

- Often they aren't: Chase, VISA, Zelle, and Federal Reserves are all just spreadsheets at the end of the day
- Centralized Blockchains add one extra level of security: create a transaction history that can't be arbitrarily tampered with: only censored or rewound.
- That's still a lot of power in the admin's hands!

A Centralized Blockchain

Centralized Blockchains are in fact widely-used:

- Binance Smart Chain
- Coinbase Base
- Ripple XRP
- "Permissioned" or "Private" Blockchains use **Proof of Authority**: the admin's public key *per se* is hard-coded into the protocol.

When are these better than just a spreadsheet?

- Often they aren't: Chase, VISA, Zelle, and Federal Reserves are all just spreadsheets at the end of the day
- Centralized Blockchains add one extra level of security: create a transaction history that can't be arbitrarily tampered with: only censored or rewound.
- That's still a lot of power in the admin's hands!

Can we take away even more power from the admin? Can we make a set of rules to agree on a shared history of transactions without needing to trust anybody?

A Decentralized Blockchain

- A natural, "democratic" rule would be to have multiple admins taking turns adding transactions to the chain

A Decentralized Blockchain

- A natural, "democratic" rule would be to have multiple admins taking turns adding transactions to the chain
- How do we prevent one admin from just pretending to be multiple admins?

A Decentralized Blockchain

- A natural, "democratic" rule would be to have multiple admins taking turns adding transactions to the chain
- How do we prevent one admin from just pretending to be multiple admins?
- Needs to be some "cost"/"barrier" to becoming an admin

A Decentralized Blockchain

- A natural, "democratic" rule would be to have multiple admins taking turns adding transactions to the chain
- How do we prevent one admin from just pretending to be multiple admins?
- Needs to be some "cost"/"barrier" to becoming an admin
- Bitcoin solves this through **proof of work**
- Ethereum solves this through **proof of stake**

Hash Puzzles

- The amount of computational power in the world is finite!
- Let's try to find a number whose SHA256 hash has a 0 at the beginning...

Hash Puzzles

- The amount of computational power in the world is finite!
- Let's try to find a number whose SHA256 hash has a 0 at the beginning...
- Why is this a nice puzzle? Like a PhD
  1. Hard to do
  2. Proves that you did something hard to do

Hash Puzzles

- The amount of computational power in the world is finite!
- Let's try to find a number whose SHA256 hash has a 0 at the beginning...
- Why is this a nice puzzle? Like a PhD
  1. Hard to do
  2. Proves that you did something hard to do
- With puzzle solution in hand, you become the admin!
- **Incentives:** solving the puzzle gives you some BTC as a block reward
- **Consensus:** everyone works on the longest chain

How to Prevent Doublespending?

- I may want to spend some money to buy a coffee, then "unspend" it (but keep my coffee!)
- Can't just selectively delete the transaction from history (why?)

How to Prevent Doublespending?

- I may want to spend some money to buy a coffee, then "unspend" it (but keep my coffee!)
- Can't just selectively delete the transaction from history (why?)
  - Hashes verify integrity of history!
- But...could go back in time, then create a parallel universe (a fork in the chain) where I never spent the money
  - This would be called a doublespend

Turning Back Time...

- Anyone can try, since anyone can try to mine and become admin
- When would I succeed?

Turning Back Time...

- Anyone can try, since anyone can try to mine and become admin
- When would I succeed?
  - Because everyone uses the **longest chain**, I must be able to hash faster than **everyone else combined**
  - Control 51% of the computing power in the Bitcoin ecosystem!
- For more details, see Stanford CS251 slides

Roadmap

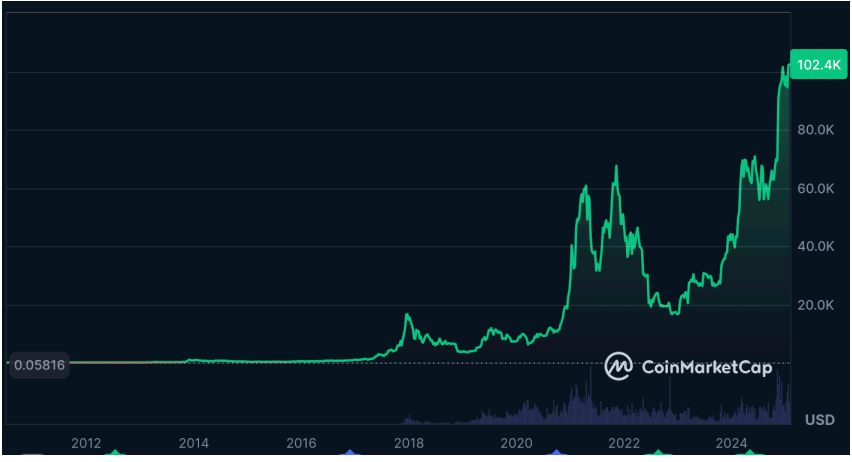
Satoshi Nakamoto

- Started sending emails about Bitcoin August 2008
- Oct 2008: Bitcoin white paper, code released
- January 9: first block mined!
  - "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks"
- May 22, 2010: Laszlo Hanyecz bought two Papa John's pizzas for 10,000 BTC, now worth $600mil USD!
- Quickly became popular among black markets, Silk Road...
- Founder Ross Ulbricht was recently pardoned by Trump

Prizes for a Starcraft Tournament in 2011

Prizes:
1st Place: *$500*
2nd Place: *$250*
3rd Place: *$150*
4th Place: *$100*
5th-8th Place: *25 BitCoins*

Bitcoin Prices



Source: CoinMarketCap

Each BTC Block is a Merkle Tree of Transactions!

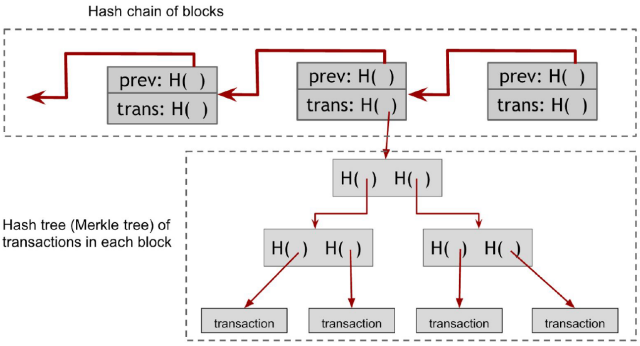


Figure 3.8. The Bitcoin block chain contains two different hash structures. The first is a hash chain of blocks that links the different blocks to one another. The second is internal to each block and is a Merkle Tree of transactions within the blocks.

- Valid block headers' hashes start with a big number of 0's (mining puzzle)

Source: NBFMG, fig. 3.8

Proof of Work Consensus

- Miners connected to each other in a network
- If you find a valid block, tell all other miners about it
  - They'll then mine the next block, based on yours
- If you solved the hash puzzle, why can't I just steal your solution?

Proof of Work Consensus

- Miners connected to each other in a network
- If you find a valid block, tell all other miners about it
  - They'll then mine the next block, based on yours
- If you solved the hash puzzle, why can't I just steal your solution?
  - My address is hashed into the block header

Proof of Work Consensus

- Miners connected to each other in a network
- If you find a valid block, tell all other miners about it
  - They'll then mine the next block, based on yours
- If you solved the hash puzzle, why can't I just steal your solution?
  - My address is hashed into the block header
- Why wouldn't I try to re-mine your block?

Proof of Work Consensus

- Miners connected to each other in a network
- If you find a valid block, tell all other miners about it
  - They'll then mine the next block, based on yours
- If you solved the hash puzzle, why can't I just steal your solution?
  - My address is hashed into the block header
- Why wouldn't I try to re-mine your block?
  - Consensus: if everyone else uses "longest chain", I do better by accepting your win, and trying to mine the next block

Proof of Work Consensus

- Notice that all properties depend on everyone "following the rules"...
  - If your block:
    - Doesn't hash to enough 0's, or
    - Steals $$, or
    - Includes invalid tx's, etc.
  - Others won't build on it, so others won't build on others not building on it,
  - So you won't get mining reward
- Simple rules are easy for a decentralized system to follow
  - By default, miners will pick the simplest possible path

Proof of Work Concensus

- If I want to send, sign a tx, and tell all the miners about it

Proof of Work Concensus

- If I want to send, sign a tx, and tell all the miners about it
  - Pile of valid signed, but not-yet in chain, tx's floating around waiting to be included by miners called the **mempool**

Proof of Work Concensus

- If I want to send, sign a tx, and tell all the miners about it
  - Pile of valid signed, but not-yet in chain, tx's floating around waiting to be included by miners called the **mempool**
- Why can't miner steal my $?

Proof of Work Concensus

- If I want to send, sign a tx, and tell all the miners about it
  - Pile of valid signed, but not-yet in chain, tx's floating around waiting to be included by miners called the **mempool**
- Why can't miner steal my $?
  - Tx specifies signature and receiver; miner can only include tx
- What if miner doesn't want to include me?

Proof of Work Concensus

- If I want to send, sign a tx, and tell all the miners about it
  - Pile of valid signed, but not-yet in chain, tx's floating around waiting to be included by miners called the **mempool**
- Why can't miner steal my $?
  - Tx specifies signature and receiver; miner can only include tx
- What if miner doesn't want to include me?
  - Someone else will, and take the tx fee...
  - ...And miner eventually has to accept my tx as valid, or "fork" whole network (since hash pointers break)

Proof of Work Concensus

- If I want to send, sign a tx, and tell all the miners about it
  - Pile of valid signed, but not-yet in chain, tx's floating around waiting to be included by miners called the **mempool**
- Why can't miner steal my $?
  - Tx specifies signature and receiver; miner can only include tx
- What if miner doesn't want to include me?
  - Someone else will, and take the tx fee...
  - ...And miner eventually has to accept my tx as valid, or "fork" whole network (since hash pointers break)
- What if I try to spend twice?

Proof of Work Concensus

- If I want to send, sign a tx, and tell all the miners about it
  - Pile of valid signed, but not-yet in chain, tx's floating around waiting to be included by miners called the **mempool**
- Why can't miner steal my $?
  - Tx specifies signature and receiver; miner can only include tx
- What if miner doesn't want to include me?
  - Someone else will, and take the tx fee...
  - ...And miner eventually has to accept my tx as valid, or "fork" whole network (since hash pointers break)
- What if I try to spend twice?
  - One tx is "invalid": doesn't obey the rules, since old tx output already spent
  - Remember, though; everything is ultimately due to consensus among miners, that "longest block of valid chains" is the legitimate blockchain

The Money Pot Game

- Two options:
  1. Get $1 for sure
  2. Fight for a $10 pot, which is equally divided between everyone who picks this option

The Money Pot Game

- Two options:
  1. Get $1 for sure
  2. Fight for a $10 pot, which is equally divided between everyone who picks this option
- Which do you pick?

The Money Pot Game

- What is going on?
  - Everyone wants the money pot
  - But the more people take the money pot, the less money there is for everyone
  - In "Nash equilibrium", the money pot becomes no better than the "outside option"

The Money Pot Game

- What is going on?
  - Everyone wants the money pot
  - But the more people take the money pot, the less money there is for everyone
  - In "Nash equilibrium", the money pot becomes no better than the "outside option"
- Similar settings: what's the equilibrium condition?
  - Firms competing for profits?

The Money Pot Game

- What is going on?
  - Everyone wants the money pot
  - But the more people take the money pot, the less money there is for everyone
  - In "Nash equilibrium", the money pot becomes no better than the "outside option"
- Similar settings: what's the equilibrium condition?
  - Firms competing for profits?
  - Going to the beach on a sunny day?

The Money Pot Game

- What is going on?
  - Everyone wants the money pot
  - But the more people take the money pot, the less money there is for everyone
  - In "Nash equilibrium", the money pot becomes no better than the "outside option"
- Similar settings: what's the equilibrium condition?
  - Firms competing for profits?
  - Going to the beach on a sunny day?
  - Getting a table for Brunch on Sunday at 12pm?

The Money Pot Game

- What is going on?
  - Everyone wants the money pot
  - But the more people take the money pot, the less money there is for everyone
  - In "Nash equilibrium", the money pot becomes no better than the "outside option"
- Similar settings: what's the equilibrium condition?
  - Firms competing for profits?
  - Going to the beach on a sunny day?
  - Getting a table for Brunch on Sunday at 12pm?
- "Equilibrium" behavior is easy to predict in such settings
  - In equilibrium, the "good" deal is so crowded that it becomes just as bad as the "bad" deal

The Money Pot Game

- What is going on?
  - Everyone wants the money pot
  - But the more people take the money pot, the less money there is for everyone
  - In "Nash equilibrium", the money pot becomes no better than the "outside option"
- Similar settings: what's the equilibrium condition?
  - Firms competing for profits?
  - Going to the beach on a sunny day?
  - Getting a table for Brunch on Sunday at 12pm?
- "Equilibrium" behavior is easy to predict in such settings
  - In equilibrium, the "good" deal is so crowded that it becomes just as bad as the "bad" deal
- Game theory language: call these congestion games, or wars of attrition, or tragedy of the commons

BTC mining

- Suppose mining a block pays 0.001 BTC
- Suppose mining block costs $60 in electricity
  - If BTC price $> \$60k$, everyone mines; blocks produced quickly
  - If BTC price $< \$60k$, nobody mines; tx's aren't included and the blockchain stops working!
- The fact that mining rate depends on BTC vs electricity price is not desirable...
- Solution: use difficulty adjustment to maintain constant block production rate

BTC Difficulty Adjustment

- Blocks should be produced every 10 minutes
- Every 2016 blocks (approx 14 days), Bitcoin considers average block production rate, and...
  - Increases/decreases hash problem difficulty, if block production is too fast/slow
- What determines how fast blocks are produced, fixing hash rate?

BTC Difficulty Adjustment

- Blocks should be produced every 10 minutes
- Every 2016 blocks (approx 14 days), Bitcoin considers average block production rate, and...
  - Increases/decreases hash problem difficulty, if block production is too fast/slow
- What determines how fast blocks are produced, fixing hash rate?
  - How many people (that is, how much electricity) is devoted to mining

BTC Difficulty Adjustment

- Blocks should be produced every 10 minutes
- Every 2016 blocks (approx 14 days), Bitcoin considers average block production rate, and...
  - Increases/decreases hash problem difficulty, if block production is too fast/slow
- What determines how fast blocks are produced, fixing hash rate?
  - How many people (that is, how much electricity) is devoted to mining
- What determines how many people mine?
  - How profitable mining is, that is, how high BTC price is, vs electricity price

BTC: Equilibrium Condition

What happens to mining when BTC price goes up?

BTC: Equilibrium Condition

What happens to mining when BTC price goes up?

- More people try to mine
- Difficulty goes up; so electricity to mine a BTC increases
- $\implies$ profit per \$ of electricity spent doesn't change

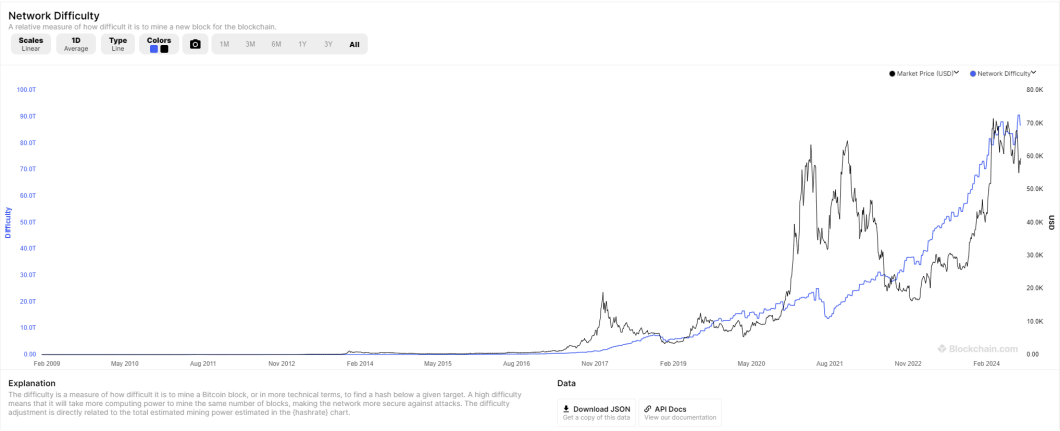
BTC: Equilibrium Condition

What happens to mining when BTC price goes up?

- More people try to mine
- Difficulty goes up; so electricity to mine a BTC increases
- $\implies$ profit per $ of electricity spent doesn't change

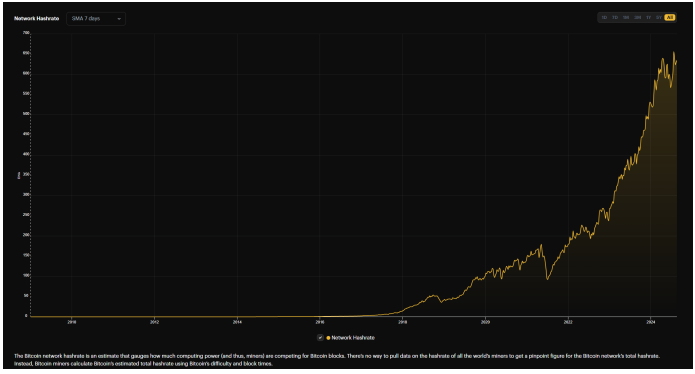
Prediction: difficulty should increase with BTC price

Difficulty vs BTC price



Source: Blockchain.com

Technological Improvement



- 842 Exahashes/s = 842 billion billion hashes a second!
- Each living person in the world could shake each other living person's hand 10 times, every second!
- $\sim 1\%$ of global energy consumption...

Source: Hashrate Index

BTC Halving

- Approximately every 4 years, issuance of BTC drops by half
- Last halving: April 2024, now 3.125 BTC per block
- Does this affect block production rate?
  - No! Difficulty adjustment always ensure a constant block production rate.

BTC Halving

- Approximately every 4 years, issuance of BTC drops by half
- Last halving: April 2024, now 3.125 BTC per block
- Does this affect block production rate?
  - No! Difficulty adjustment always ensure a constant block production rate.
- Also implies there will only ever be 21 million BTC in existence!
  - By approximately 2140, the block reward will drop to less than the smallest unit of bitcoin ($1/10^8$) and become equal to zero.
  - Difficulty adjustment ensures that block production rate is approximately constant

Transaction Fees

- Block rewards are not the only incentive to process transactions: miners also charge transaction fees to include a transaction in their blocks.

Transaction Fees

- Block rewards are not the only incentive to process transactions: miners also charge transaction fees to include a transaction in their blocks.
- Transaction fees are quoted in "sat/vB", or "Satoshis per virtual Byte"
  - A Satoshi is $1/10^8$ BTC, the smallest unit that can be transacted

Transaction Fees

- Block rewards are not the only incentive to process transactions: miners also charge transaction fees to include a transaction in their blocks.
- Transaction fees are quoted in "sat/vB", or "Satoshis per virtual Byte"
  - A Satoshi is $1/10^8$ BTC, the smallest unit that can be transacted
  - The Size of the transaction depends mostly on its format. The current most efficient format for submitting transactions was created in 2017.

Transaction Fees

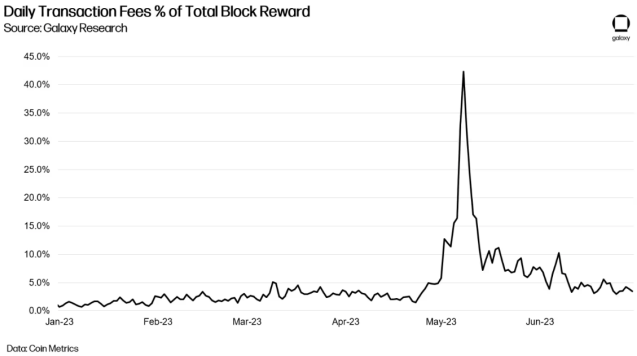
- Block rewards are not the only incentive to process transactions: miners also charge transaction fees to include a transaction in their blocks.
- Transaction fees are quoted in "sat/vB", or "Satoshis per virtual Byte"
  - A Satoshi is $1/10^8$ BTC, the smallest unit that can be transacted
  - The Size of the transaction depends mostly on its format. The current most efficient format for submitting transactions was created in 2017.
- Miners prioritize including transactions that offer higher fee rates
- As of 2/4/25, the going rate to complete your transaction within 30 minutes is 3 sat/vB, or $\$0.42$ for an average transaction

Transaction Fees

- Block rewards are not the only incentive to process transactions: miners also charge transaction fees to include a transaction in their blocks.
- Transaction fees are quoted in "sat/vB", or "Satoshis per virtual Byte"
  - A Satoshi is $1/10^8$ BTC, the smallest unit that can be transacted
  - The Size of the transaction depends mostly on its format. The current most efficient format for submitting transactions was created in 2017.
- Miners prioritize including transactions that offer higher fee rates
- As of 2/4/25, the going rate to complete your transaction within 30 minutes is 3 sat/vB, or $\$0.42$ for an average transaction
- A **"mempool"** is a pool of transactions waiting to be added to the bitcoin blockchain

Explore mempools at https://mempool.space/

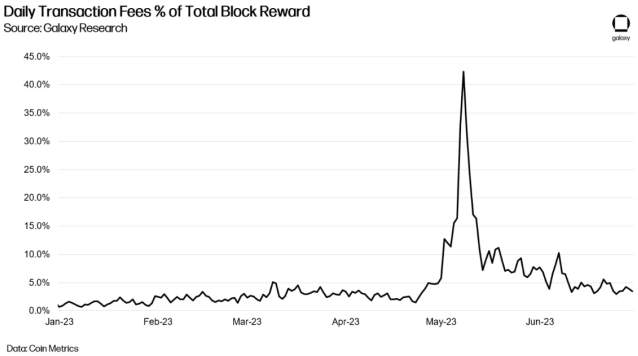
Transaction Fees as % of Block Revenue



Source: Galaxy

In the long run, this will go to 100%! Why?

Transaction Fees as % of Block Revenue



Source: Galaxy

In the long run, this will go to 100%! Why? Mining rewards will reach zero, and the only incentive to mine a block will be to collect transaction fees.

Incentives

- A centralized authority can "censor", i.e. not include tx's
- Decentralization and competition "ensure" performance
  - If you don't include my tx, the next miner will
  - Even if the US bans US miners from including tx's, someone else will include them

Further Reading on Mining Industry

- Galaxy digital 2023 and 2024
- Cool paper on mining pools

Roadmap

Smart Contracts

"A smart contract is essentially an automated agent that lives on the Ethereum network, has an Ethereum address and balance, and can send and receive transactions. A contract is "activated" every time someone sends a transaction to it, at which point it runs its code, perhaps modifying its internal state or even sending some transactions, and then shuts down."
-Ethereum Whitepaper, Vitalik Buterin (2014)

Smart Contracts

- A smart contract is:
  - A wallet, which can hold Ether (ETH),
  - Functions it can call when requested as determined by its code,
  - (optionally) a source of data to which it can read and write

Smart Contracts

- A smart contract is:
  - A wallet, which can hold Ether (ETH),
  - Functions it can call when requested as determined by its code,
  - (optionally) a source of data to which it can read and write
- If you've heard of object oriented programming: smart contracts are a bundle of functions (code), data, as well as ETH balances

Smart Contracts

- A smart contract is:
  - A wallet, which can hold Ether (ETH),
  - Functions it can call when requested as determined by its code,
  - (optionally) a source of data to which it can read and write
- If you've heard of object oriented programming: smart contracts are a bundle of functions (code), data, as well as ETH balances
- The most popular coding language which compiles Ethereum script code (a minimal Turing-complete language akin to assembly) is called **Solidity**

Smart Contracts

- A smart contract is:
  - A wallet, which can hold Ether (ETH),
  - Functions it can call when requested as determined by its code,
  - (optionally) a source of data to which it can read and write
- If you've heard of object oriented programming: smart contracts are a bundle of functions (code), data, as well as ETH balances
- The most popular coding language which compiles Ethereum script code (a minimal Turing-complete language akin to assembly) is called **Solidity**

Deploying a smart contract or calling its functions requires a **gas fee**: ETH must be paid to the network to run the code.

Smart Contract Example: Tokens

- An ETH address, "technically", only "holds" Ether ("ETH") natively – no other tokens!
- An ETH address can be assigned non-ETH tokens through smart contracts
- A smart contract can define tokens using the **ERC-20** standard
  - So-called because it was originally the 20th submission to "Ethereum Requests for Comments"

Unlike BTC, ETH has native gas fees

"In Bitcoin, there are no mandatory transaction fees. Transactions can optionally include fees which are paid to miners, and it is up to the miners to decide what fees they are willing to accept. In Bitcoin, such a mechanism is already imperfect...In Ethereum, because of its Turing-completeness, a purely voluntary fee system would be catastrophic. Instead, Ethereum will have a system of mandatory fees, including a transaction fee and six fees for contract computations."
-Ethereum Whitepaper, Vitalik Buterin (2014)

Ethereum Virtual Machine Native Gas Costs

evm.codes

Proof of Stake

- Originally, ETH mining proof-of-work, similar to BTC
- On Sept 15 2022, ETH switched to proof of stake – called **"The Merge"**
- Reduced energy consumption by **over 99%**!
- Users deposit ("stake") 32 ETH into a special smart contract
- Stakers randomly selected (prop. to stake) to propose new blocks
- If a staker is caught trying to do "bad things" (e.g. propose 2 inconsistent blocks), they are "slashed": staked ETH is removed by smart contract
- Most newer blockchains (SOL, LUNA, AVAX) use proof-of-stake

Capacity of Ethereum

- Ethereum currently processes around 15 transactions per second
- This is pretty bad! Visa is 24,000 tps!
  - Every ETH staker has to run every transaction, and keep a copy of all data!
- Decentralization very expensive, in terms of computational efficiency!
- Proof-of-stake doesn't help throughput
  - Rollups do, which we will cover

Roadmap

Rollups or "L2's"

ETH is getting expensive! One set of solutions is rollups (also called L2's). Idea:

- Put our ETH, tokens, etc. in a "smart contract pot" (L2 bridge)
- People do transactions (smart contract, etc.) using funds in the pot, without doing anything on "main chain"
- Transactions are periodically batch settled

Rollups or "L2's"

ETH is getting expensive! One set of solutions is rollups (also called L2's). Idea:

- Put our ETH, tokens, etc. in a "smart contract pot" (L2 bridge)
- People do transactions (smart contract, etc.) using funds in the pot, without doing anything on "main chain"
- Transactions are periodically batch settled
- Need to guarantee whatever happens on "rollup" is exactly what would happen on ETH main chain! Two methods:
  - **Optimistic rollup** (Optimism, Arbitrum, Coinbase Base): Can "challenge" L2 transactions, will (essentially) run on L1 if challenged
  - **ZK-rollup** (zkSync, StarkWare, Polygon): Some math black magic proves validity of transactions

Rollups or "L2's"

ETH is getting expensive! One set of solutions is rollups (also called L2's). Idea:

- Put our ETH, tokens, etc. in a "smart contract pot" (L2 bridge)
- People do transactions (smart contract, etc.) using funds in the pot, without doing anything on "main chain"
- Transactions are periodically batch settled
- Need to guarantee whatever happens on "rollup" is exactly what would happen on ETH main chain! Two methods:
  - **Optimistic rollup** (Optimism, Arbitrum, Coinbase Base): Can "challenge" L2 transactions, will (essentially) run on L1 if challenged
  - **ZK-rollup** (zkSync, StarkWare, Polygon): Some math black magic proves validity of transactions
- Industry state:
  - $30B locked! ETH market cap is $408B, stables ~$141B
  - Optimistic rollups somewhat bigger than ZKs