What is a blockchain?

Abstract answer: a blockchain provides coordination between many parties, when there is no single trusted party.

if trusted party exists ⇒ no need for a blockchain

[financial systems: often no trusted party]
Blockchains: what is the new idea?

2009

Bitcoin

Several innovations:

• A practical **public append-only data structure**, secured by **replication** and **incentives**

• A fixed supply asset (**BTC**). Digital payments, and more.
Several innovations:

- **Blockchain computer**: a fully programmable environment
  \[\Rightarrow\text{public programs that manage digital and financial assets}\]

- **Composability**: applications running on chain can call each other
Blockchains: what is the new idea?

- 2009: Bitcoin
- 2015: Ethereum
- 2017-2022: growth of DeFi, NFTs, DAOs
So what is this good for?

(1) Basic application: a digital currency (stored value)
  • Current largest: Bitcoin (2009), Ethereum (2015)
  • Global: accessible to anyone with an Internet connection

Opinion

Bitcoin Has Saved My Family

“Borderless money” is more than a buzzword when you live in a collapsing economy and a collapsing dictatorship.

By Carlos Hernández
Mr. Hernández is a Venezuelan economist.

Feb. 23, 2019
What else is it good for?

(2) Decentralized applications (DAPPs)
  • **DeFi**: financial instruments managed by public programs
    • examples: stablecoins, lending, exchanges, ....
  • **Asset management** (NFTs): art, game assets, domain names.
  • **Decentralized organizations** (DAOs): (decentralized governance)
    • DAOs for investment, for donations, for collecting art, etc.

(3) New programming model: writing decentralized programs
## Assets managed by DAPPs

<table>
<thead>
<tr>
<th>DAPP</th>
<th>Platform</th>
<th>Product</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MakerDAO</td>
<td>Ethereum</td>
<td>StableCoin</td>
<td>$7.30B</td>
</tr>
<tr>
<td>Curve</td>
<td>Ethereum</td>
<td>Exchange</td>
<td>$4.60B</td>
</tr>
<tr>
<td>Aave</td>
<td>Ethereum</td>
<td>Lending</td>
<td>$4.09B</td>
</tr>
<tr>
<td>Uniswap</td>
<td>Ethereum</td>
<td>Exchange</td>
<td>$3.73B</td>
</tr>
<tr>
<td>Compound</td>
<td>Ethereum</td>
<td>Lending</td>
<td>$2.23B</td>
</tr>
</tbody>
</table>

Sep. 2022
<table>
<thead>
<tr>
<th>Cryptocurrency</th>
<th>24h Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitcoin (BTC)</td>
<td>$26.6B</td>
</tr>
<tr>
<td>Ethereum (ETH)</td>
<td>$12.1B</td>
</tr>
<tr>
<td>Solana (SOL)</td>
<td>$0.88B</td>
</tr>
</tbody>
</table>

**Sep. 2022**
# Active developers since launch (as of 12/31/2021)

![Graph showing the growth of active developers on various blockchain projects since launch, with Ethereum and Bitcoin highlighted.](source: electric capital)
Central Bank Digital Currency (CBDC)

China Moves Forward With National Digital Currency

by Sam Klebanov  — September 3, 2021
What is a blockchain?

- **Data Availability / Consensus Layer**
  - Sequencer: orders transactions

- **Execution engine** (blockchain computer)

- **Applications** (DAPPs, smart contracts)

- **User facing tools** (cloud servers)
A **public** append-only data structure:

- **Persistence**: once added, data can never be removed*
- **Safety**: all honest participants have the same data**
- **Liveness**: honest participants can add new transactions
- **Open(?)**: anyone can add data (no authentication)

Data Availability / Consensus layer achieved by replication
How are blocks added to chain?

blockchain

sk_A  signed

6.25 BTC

verify block

I am the leader

verify block

sk_B  signed

sk_C  signed
How are blocks added to chain?
Why is consensus a hard problem?

The good case: copies are consistent
Why is consensus a hard problem?

Problems:
- Network delays

\[\Delta\text{-delay}\]

\(\text{can affect Tx order}\)
Why is consensus a hard problem?

Problems:
- Network delays
- Network partition
Why is consensus a hard problem?

Problems:
- crash

Tx1, Tx2, Tx4

crashed

Tx3??

Tx1

Tx2

Tx4

Tx1, Tx2, Tx4

Tx1, Tx2, Tx4
Why is consensus a hard problem?

Problems:
- crash
- malice

Consensus protocols: next week
Decentralized applications (DAPPs):

- Run on blockchain: code and state are written on chain
- Accept Tx from users ⇒ state transitions are recorded on chain
Next layer: the blockchain computer

Top layer: user facing servers

on-chain state

blockchain computer

Data availability / Consensus layer
Lots of experiments:
This course

Cryptography

Economics

Distributed systems
Course organization

1. The starting point: Bitcoin mechanics
2. Consensus protocols
3. Ethereum and decentralized applications
4. DeFi: decentralized applications in finance
5. Private transactions on a public blockchain (SNARKs and zero knowledge proofs)
6. Scaling the blockchain: getting to 10K Tx/sec
7. Interoperability among chains: bridges and wrapped coins
Course organization

cs251.stanford.edu

• Homework problems, projects, final exam
• Optional weekly sections on Friday

Please tell us how we can improve ...
Don’t wait until the end of the quarter
Let’s get started ...
(1) cryptographic hash functions

An efficiently computable function \( H : M \rightarrow T \)
where \( |M| \gg |T| \)

\[ T = \{0,1\}^{256} \]
**Collision resistance**

**Def:** a collision for $H: M \to T$ is pair $x \neq y \in M$ s.t. $H(x) = H(y)$

$|M| \gg |T|$ implies that many collisions exist

**Def:** a function $H: M \to T$ is collision resistant if it is “hard” to find even a single collision for $H$ (we say $H$ is a CRF)

**Example:** SHA256: $\{x : \text{len}(x) < 2^{64} \text{ bytes}\} \to \{0,1\}^{256}$

(output is 32 bytes)
Application: committing to data on a blockchain

Alice has a large file $m$. She posts $h = H(m)$ \hspace{1cm} (32 bytes)

Bob reads $h$. Later he learns $m'$ s.t. $H(m') = h$

$H$ is a CRF $\Rightarrow$ Bob is convinced that $m' = m$

(otherwise, $m$ and $m'$ are a collision for $H$)

We say that $h = H(m)$ is a **binding commitment** to $m$

(note: not hiding, $h$ may leak information about $m$)
committing to a list (of transactions)

Alice has \( S = (m_1, m_2, ..., m_n) \)

**Goal:**
- Alice posts a **short** binding commitment to \( S \), \( h = \text{commit}(S) \)
- Bob reads \( h \). Given \((m_i, \text{proof } \pi_i)\) can check that \( S[i] = m_i \)

Bob runs \( \text{verify}(h, i, m_i, \pi_i) \rightarrow \text{accept/reject} \)

**security:** adv. cannot find \((S, i, m, \pi)\) s.t. \( m \neq S[i] \) and

\( \text{verify}(h, i, m, \pi) = \text{accept} \) where \( h = \text{commit}(S) \)
Merkle tree (Merkle 1989)

Goal:
• commit to list $S$ of size $n$
• Later prove $S[i] = m_i$

commitment

$commitment$

$h$

Merkle tree

commitment

$m_1$ $m_2$ $m_3$ $m_4$ $m_5$ $m_6$ $m_7$ $m_8$

list of values $S$
Merkle tree (Merkle 1989) [simplified]

Goal:
- commit to list $S$ of size $n$
- Later prove $S[i] = m_i$

To prove $S[4] = m_4$,
proof $\pi = (m_3, y_1, y_6)$

length of proof: $\log_2 n$
To prove $S[4] = m_4$, proof $\pi = (m_3, y_1, y_6)$

Bob does:
\[
\begin{align*}
  y_2 &\leftarrow H(m_3, m_4) \\
  y_5 &\leftarrow H(y_1, y_2) \\
  h' &\leftarrow H(y_5, y_6) \\
\end{align*}
\]
accept if $h = h'$
**Thm:** For a given $n$: if $H$ is a CRF then

```
adv. cannot find $(S, i, m, \pi)$ s.t. $|S| = n, \quad m \neq S[i],$
```

```
h = \text{commit}(S), \text{ and } \text{verify}(h, i, m, \pi) = \text{accept}
```

(to prove, prove the contra-positive)

**How is this useful?** To post a block of transactions $S$ on chain suffices to only write $\text{commit}(S)$ to chain. Keeps chain small.

\[ \Rightarrow \text{ Later, can prove contents of every Tx.} \]
Merkle proofs are used to prove that a Tx is “on the block chain”
Another application: proof of work

**Goal:** computational problem that
- takes time $\Omega(D)$ to solve, but
- solution takes time $O(1)$ to verify

(D is called the **difficulty**)

How? \( H: X \times Y \rightarrow \{0,1,2, \ldots, 2^n - 1\} \)  e.g.  \( n = 256 \)

- puzzle: input \( x \in X \), output \( y \in Y \) s.t. \( H(x, y) < 2^n/D \)
- verify\((x, y)\): accept if \( H(x, y) < 2^n/D \)
**Another application: proof of work**

**Thm:** if $H$ is a “random function” then the best algorithm requires $D$ evaluations of $H$ in expectation.

Note: this is a parallel algorithm  
$\Rightarrow$ the more machines I have, the faster I solve the puzzle.

Proof of work is used in some consensus protocols (e.g., Bitcoin)  
Bitcoin uses $H(x, y) = \text{SHA256}(\text{SHA256}(x.y))$
Cryptography background:
Digital Signatures

How to authorize a transaction
Signatures

Physical signatures: bind transaction to author

Bob agrees to pay Alice 1$

Bob agrees to pay Alice 100$

Problem in the digital world:

anyone can copy Bob’s signature from one doc to another
**Digital signatures**

Solution: make signature depend on document

Bob agrees to pay Alice $1

Verifier

'Signature' or 'reject'

Verifier

public verification key (pk)

Signer

secret signing key (sk)

signature

signing algorithm
Digital signatures: syntax

Def: a signature scheme is a triple of algorithms:

• **Gen()**: outputs a key pair \((pk, sk)\)
• **Sign\((sk, msg)\)** outputs sig. \(\sigma\)
• **Verify\((pk, msg, \sigma)\)** outputs ‘accept’ or ‘reject’

Secure signatures: (informal)

Adversary who sees signatures on many messages of his choice, cannot forge a signature on a new message.
Families of signature schemes

1. RSA signatures (old ... not used in blockchains):
   - long sigs and public keys (∑256 bytes), fast to verify

2. Discrete-log signatures: Schnorr and ECDSA (Bitcoin, Ethereum)
   - short sigs (48 or 64 bytes) and public key (32 bytes)

3. BLS signatures: 48 bytes, aggregatable, easy threshold
   (Ethereum 2.0, Chia, Dfinity)

4. Post-quantum signatures: long (∑600 bytes)
   details in CS255
Signatures on the blockchain

Signatures are used everywhere:
- ensure Tx authorization,
- governance votes,
- consensus protocol votes.
Next lecture: the Bitcoin blockchain