Decentralized Finance

Introduction to Blockchain technology

Instructors: Dan Boneh, Arthur Gervais, Andrew Miller, Christine Parlour, Dawn Song
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]
What is a blockchain?

- **consensus layer**
- **compute layer** (blockchain computer)
- **applications** (DAPPs, smart contracts)
- **user facing tools** (cloud servers)
Consensus layer  (informal – not the topic of this course)

A public append-only data structure:

• **Persistence**: once added, data can never be removed*

• **Consensus**: all honest participants have the same data**

• **Liveness**: honest participants can add new transactions

• **Open(?)**: anyone can add data

Layer 1:  consensus layer

* achieved by replication

**
How are blocks added to chain?

blockchain

signed

2 ETH

I am the leader

verify block

skₐ

skₖ

skₖ₃
How are blocks added to chain?

blockchain

\[
\text{sk}_A \\
\text{sk}_B \\
\text{sk}_C
\]

I am the leader

2 ETH
DAPP logic is encoded in a program that runs on blockchain

- Rules are enforced by a public program (public source code)
  ⇒ transparency: no single trusted 3rd party

- The DAPP program is executed by parties who create new blocks
  ⇒ public verifiability: everyone can verify state transitions
Apps layer: Decentralized applications (DAPPs)

- Run on blockchain computer

- applications (DAPPs, smart contracts)

- blockchain computer

- consensus layer
UI Layer: Common DAPP architecture

UI Layer: user facing servers

end user

DAPP

on-chain state

DAPP

blockchain computer

DAPP

consensus layer
[source: the Block Genesis]
lots of experiments ...

<table>
<thead>
<tr>
<th>Name</th>
<th>Chain</th>
<th>Category</th>
<th>Locked (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aave</td>
<td>Multichain</td>
<td>Lending</td>
<td>$15.63B</td>
</tr>
<tr>
<td>Curve Finance</td>
<td>Multichain</td>
<td>DEXes</td>
<td>$10.71B</td>
</tr>
<tr>
<td>InstaDAp</td>
<td>Ethereum</td>
<td>Lending</td>
<td>$10.66B</td>
</tr>
<tr>
<td>Compound</td>
<td>Ethereum</td>
<td>Lending</td>
<td>$10.43B</td>
</tr>
<tr>
<td>Maker</td>
<td>Ethereum</td>
<td>Lending</td>
<td>$9.11B</td>
</tr>
<tr>
<td>Uniswap</td>
<td>Ethereum</td>
<td>DEXes</td>
<td>$7.29B</td>
</tr>
<tr>
<td>Convex Finance</td>
<td>Ethereum</td>
<td>Assets</td>
<td>$5.59B</td>
</tr>
</tbody>
</table>
Let’s get started ...

Next segment: cryptographic background

See you there
Cryptographic Background: hash functions

https://defi-learning.org/
(1) cryptographic hash functions

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

Megabytes $\rightarrow$ 32 bytes

Hash value $T = \{0,1\}^{256}$
Collision resistance

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

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

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

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

Details in crypto MOOC
An application: committing to data

Alice has a large file $m$. She publishes $h = H(m)$ (32 bytes)

Bob has $h$. Later Alice sends $m'$ s.t. $H(m') = h$

$H$ is a CRH $\Rightarrow$ 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 publishes a short binding commitment to \( S \), \( h = \text{commit}(S) \)
- Bob has \( h \). 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

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

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

commitment

\[ h \]

Merkle tree commitment

\[ m_1 \quad m_2 \quad m_3 \quad m_4 \quad m_5 \quad m_6 \quad m_7 \quad m_8 \]

list of values  S
Merkle tree  (Merkle 1989)

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 \( \pi \): \( \log_2 n \)
Merkle tree  (Merkle 1989)

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

Bob does:

$y_2 \leftarrow H(m_3, m_4)$
$y_5 \leftarrow H(y_1, y_2)$
$h' \leftarrow H(y_5, y_6)$
accept if $h = h'$
Merkle tree  (Merkle 1989)

**Thm:** $H$ CRH $\Rightarrow$ adv. cannot find $(S, i, m, \pi)$ s.t. $m \neq S[i]$ and verify$(h, i, m, \pi) = \text{accept}$ where $h = \text{commit}(S)$

(to prove, prove the contra-positive)

How is this useful? Super useful.   Example:

- When writing a block of transactions $S$ to the blockchain, suffices to write $\text{commit}(S)$ to chain.   Keep chain small.
- Later, can prove contents of every Tx.
Abstract block chain

Merkle proofs are used to prove that a Tx is “on the block chain”
Next segment: digital signatures

How to authorize transactions??
Cryptographic Background: Digital Signatures

https://defi-learning.org/
Digital Signatures

- In the last segment we looked at cryptographic hash functions.
- In this segment we will look at digital signatures:

  how to approve 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

'accept' or 'reject'

Signer

secret signing key (sk)

Verifier

public verification key (pk)

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 pk and sigs on many messages of her choice, cannot forge a signature on a new message.
Families of signature schemes

1. **RSA signatures (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 keys (32 bytes)

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

4. **Post-quantum signatures:** long (≥768 bytes)
Signatures on the blockchain

Signatures are used everywhere:
- ensure Tx authorization,
- governance votes,
- consensus protocol votes.

\[ \text{verify Tx} \]

\[ \text{verify Tx} \]

\[ \text{verify Tx} \]
We covered two important cryptographic primitives:

1. Collision resistant hash functions and Merkle trees,
2. Digital signatures.

Another important cryptographic primitive is a **SNARK proof**:  
- Used for scaling and privacy  
- We will discuss SNARKs in detail in the lecture on privacy
Next segment: scaling the blockchains

Can we make it fast??
Scaling Blockchains

https://defi-learning.org/
Scaling

Transaction rates (Tx/sec):

- Bitcoin: can process about $5$ (Tx/sec)
- Ethereum: can process about $20$ (Tx/sec)

Tx Fees fluctuate: $2$ to $60$ for simple Tx
Ethereum Tx fees (gas prices)

Average Transaction Fee Chart
Source: Etherscan.io

$68
Scaling

Transaction rates (Tx/sec):

- Bitcoin: can process about 5 (Tx/sec)
- Ethereum: can process about 20 (Tx/sec)
- The visa network: can process up to 24,000 (Tx/sec)

Tx Fees fluctuate: 2$ to 60$ for simple Tx

Can we scale blockchains to visa speeds? ... with low Tx fees
Scaling approaches

Many approaches to scaling blockchains:

- **Faster consensus**: modern blockchains (e.g., Solana, Polkadot, Avalanche, ...)
- **Payment channels**: most Tx are off chain Peer-to-Peer (e.g., Lightening)
- **Layer 2 approaches**:
  - **zkRollup, optimistic Rollup**: batch many Tx into a single Tx
- **Sidechains**: Polygon and others
- **many other ideas ...**
(1) Payment channels (high level idea)

blockchain

$100 held in channel (e.g., UTXO or DAPP)

Alice creates payment channel to Bob: value $100

verify channel created correctly

HTLC logic: Hashed TimeLock Contract
Two ways to close channel:
• Tx with Alice sig: can close channel after 30 days, or
• Tx with Alice sig & Bob sig: close channel right away
Payment channels (high level idea)

blockchain

$100 held in channel (e.g., UTXO)

Bob can sign Tx and close channel
… but he would rather wait (up to 30 days)

Tx: distribute funds: Alice: 95; Bob: 5

( off chain message! )
(1) Payment channels (high level idea)

$100$ held in channel (e.g., UTXO)

another payment: pay Bob: $15$

Tx: distribute funds: Alice: $80$; Bob: $20$  \[\text{sig}_{\text{Alice}}\]
(1) Payment channels (high level idea)

blockchain

$100 \text{ held in channel (e.g., UTXO)}$

Alice  \hspace{1cm} another payment: pay Bob: 10$

Tx: distribute funds: Alice: 70; Bob: 30  \hspace{1cm} \text{sig}_{\text{Alice}}$

Bob
(1) Payment channels (high level idea)

- Blockchain
- $100 held in channel (e.g., UTXO)
- Alice: 70
- Bob: 30
- Either side can close channel (Alice only after 30 days)
- Tx: distribute funds: Alice: 70; Bob: 30
- Sig_Alice
- Tx, Sig_A, Sig_B
(1) Payment channels (high level idea)

blockchain

$100 held in channel (e.g., UTXO)

Alice: 70
Bob: 30

either side can close channel (Alice only after 30 days)

main point: participants only touch chain when a channel is created or closed.

Bi-directional channels are also possible.

Tx, sig_A, sig_B
Payment networks

Lots of bi-directional payment channels

Alice pays Bob by finding the cheapest route through the network
⇒ while channels are open, nothing touches the blockchain
The case of El Salvador

Payment channels are necessary to enable state-wide adoption

- **Strike wallet:** connects to the Bitcoin Lightening network
(2) Scaling Ethereum Using Rollup
(2) Scaling Ethereum Using Rollup

Main tool: SNARK (much more on SNARKs later)

C: a program that always terminates in $\leq B$ steps

x: public input to C, $w$: private input to C

(C, x, w) \quad \text{short proof } \pi \quad (C, x)

prover \quad \text{verifier}
(2) Scaling Ethereum Using Rollup

Main point:
Verifier’s run time is *much* less than running C

I am convinced prover knows w s.t. \( C(x, w) = 1 \)

C: a program that always terminates in \( \leq B \) steps

x: public input to C,
w: private input to C

(prover, C, x, w) → short proof \( \pi \) → (C, x) → verifier

More on SNARKs later.
(2) Rollup: zk and optimistic

Standard L1 chains: every miner must verify every posted Tx

Rollup coordinator: compresses a thousand Tx into one on-chain proof (SNARK)
zkRollup (simplified)

Alice:
5 DAI
3 ETH

Bob:
2 ETH

Zoe:
1 ETH
3 BAT

atomic swap:
[A → B: 2 ETH], \( \sigma_A \)

[B → Z: 1 ETH]

[Z → B: 2 BAT]

\( \sigma_B \), \( \sigma_Z \)

Layer 1 blockchain (e.g. Ethereum)

block 354

Merkle Tree

Alice: 5 DAI
3 ETH

Bob: 2 ETH

... 

Zoe: 1 ETH
3 BAT

rollup coordinator
zkRollup (simplified)

Alice: 5 DAI, 1 ETH
Bob: 3 ETH, 2 BAT
Zoe: 2 ETH, 1 BAT

Layer 1 blockchain (e.g. Ethereum)

Block 354
Block 357

Merkle Tree

[A→B: 2 ETH, \(sig_A\)]
[B→Z: 1 ETH]
[Z→B: 2 BAT, \(sig_B\) \(sig_Z\)]
atomic swap:

\(\text{Tx summary, \text{[SNARK]}}\)
Transferring assets to and from L2

- **Transactions within a Rollup system are easy:**
  - Batch settlement on L1 network (e.g., Ethereum)

- **Moving funds in to or out of Rollup system (L1 ⇔ L2) is more expensive:**
  - Requires posting more data on L1 network  \(\Rightarrow\) higher Tx fees.

- **Moving funds from one Rollup system to another (L2 ⇔ L2):**
  - Either via L1 network (expensive), or via a direct L2 ⇔ L2 bridge (cheap)
Migrating a project from L1 Ethereum to L2 zkRollup

Upcoming development: **zkEVM** (e.g., MatterLabs and others).

**Solidity compatibility:**
- Coordinator can produce a SNARK proof for the execution of a short Solidity program:
  - easy to migrate a DAPP from L1 Ethereum to L2 zkRollup.
  - reduced Tx fees and increased Tx rate compared to L1
Optimistic Rollup (simplified)  [e.g., Optimism, Arbitrum]

Same principle as zkRollup, but no SNARK proof

Instead: coordinator posts Tx data on chain **without** a proof
- Then give a few days for validators to complain:
  if a posted Tx is invalid $\Rightarrow$
  anyone can submit a **fraud proof** and win a reward,
  Rollup server gets slashed.

Benefit: simple full EVM compatibility, less work for server.
Data availability: zkSync vs. zkPorter

Is the coordinator a central point of failure? (centralization fears??)

Answer: No!

coordinator fails $\Rightarrow$ users find another coordinator to produce proofs

- Complication: new coordinator needs all current account information
  - How to get the data if the old coordinator is dead?

- Two solutions: zkSync and zkPorter. They work concurrently.
Data availability: zkSync vs. zkPorter

- **zkSync**: store all Tx summaries on the L1 blockchain (Ethereum)
  - L1 chain accepts Tx batch only if it includes summary of all Tx
  - Other coordinators can reconstruct L2 state from L1 blockchain
  - Downside: higher Ethereum Tx fees. Good for high value assets

- **zkPorter**: store Tx data on a new blockchain
  - maintained by a set of staked coordinators
  - Cheap off-chain storage, but lower guarantee than zkSync

- Customer can choose how coordinator will store its account.
That’s it on this topic …

Next segment: interoperability

How to move assets from one chain to another
Interchain Interoperability

https://defi-learning.org/
Can I use Serum??

20 DOT
Interoperability

- **Interoperability:**
  - a user owns funds or assets on one blockchain system.
    Goal: enable the user to move funds and/or assets to another system.

- **Composability:**
  - enable a DAPP on one blockchain to call a DAPP on another

Both are easy if the entire world used Ethereum
- In reality: many blockchain systems that need to interoperate
- Several cross-chain protocols: XCMP, IBC, ...
How to move assets? Building a federated bridge (simplified)

user

1₿

bridge address

Veriﬁed (signed)

bridge contract

mint one pegged- 1 P₿

to use in DeFi

user

1 P₿

staked validators

signing keys
How to move assets? Building a federated bridge (simplified)

Why external validators? Bridge contract cannot store Bitcoin signing key

1₿

bridge address

1 P฿

bridge contract

1 P฿

user

user

signing keys

staked validators

Bitcoin Tx (signed)
End of lecture: quick review

Cryptographic primitives:
- Hash functions: committing to large amounts of data
- Digital signatures: authorizing actions

Scaling the blockchain
- Payment channels and Rollups

Interoperability: via bridges and pegged coins.
END OF TOPIC

https://defi-learning.org/