Invited talk final lecture. Final exam will be released this week.
Quick Recap: zkRollup

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

Merkle Tree
root

atomic swap:
[B→Z: 1 ETH]
[Z→B: 2 BAT]

rollup server

L1 blockchain (e.g. Ethereum)
block 354

[A→B: 2 ETH], $\text{sig}_A$

Tx

[A→B: 2 ETH], $\text{sig}_A$
Quick Recap: zkRollup

rollable server

new root

Merkle Tree

Alice: 5 DAI
1 ETH

Bob: 3 ETH
2 BAT

Zoe: 2 ETH
1 BAT

atomic swap:

[A→B: 2 ETH], \( sig_A \)

[B→Z: 1 ETH]

[Z→B: 2 BAT]

\( sig_B \) \( sig_z \)

L1 blockchain (verifies SNARK)

block 354

block 357

Tx data, SNARK

[A→B: 2 ETH], \( sig_A \)

Berverage

new root
Key points

The Rollup server stores all account balances
  • L1 chain does not store explicit balances

\{Rollup: Tx data written to L1 chain (16 gas per byte)
   Validium: Tx data written to off-chain staked servers (cheaper)\}

why store Tx data? ... backup in case rollup server fails

Can we hide Tx data from the Rollup server and the public?
  • Yes! Using (zk)^2-SNARKs
A brief discussion of NFTs
NFTs: managing digital assets

Example digital assets: (ERC-721)

- Digital art: opensea, foundation
- Collector items: NBA top shots
- Game items: horses (zed.run), axies, ...
- Metaverse: ENS, plots in a virtual land

Why manage on a blockchain? Why not manage centrally?

- Blockchain ensures long-term ownership, until sale.
- Provides a trusted record of provenance (forgeries are evident)
Example: CryptoPunks


All offers and sales recorded on Ethereum (250 lines of Solidity)

<table>
<thead>
<tr>
<th>Bid</th>
<th>Seller</th>
<th>Price</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>beautiful...</td>
<td>150Ξ ($497,239)</td>
<td>Aug 24, 2021</td>
</tr>
<tr>
<td>Sold</td>
<td>gmoney</td>
<td>49.50Ξ ($149,939)</td>
<td>Aug 18, 2021</td>
</tr>
<tr>
<td>Bid</td>
<td>0xa04e64</td>
<td>49.50Ξ ($149,024)</td>
<td>Aug 18, 2021</td>
</tr>
<tr>
<td>Sold</td>
<td>gr8wxl</td>
<td>21Ξ ($31,117)</td>
<td>Mar 05, 2021</td>
</tr>
<tr>
<td>Offered</td>
<td>0x84c920</td>
<td>21Ξ ($31,117)</td>
<td>Mar 05, 2021</td>
</tr>
<tr>
<td>Sold</td>
<td>0x02751f</td>
<td>0.30Ξ ($67)</td>
<td>Aug 03, 2017</td>
</tr>
<tr>
<td>Offered</td>
<td>gr8wxl</td>
<td>0.30Ξ ($59)</td>
<td>Jul 30, 2017</td>
</tr>
<tr>
<td>Claimed</td>
<td>0x02751f</td>
<td></td>
<td>Jun 23, 2017</td>
</tr>
</tbody>
</table>

https://www.larvalabs.com/cryptopunks/details/7610
The resulting gas wars

Gas prices spike around highly-anticipated NFT launches:
... maybe don’t use first come first serve??

https://www.paradigm.xyz/2021/10/a-guide-to-designing-effective-nft-launches/
NFTs are about managing ownership of general digital assets

Growing list of categories on OpenSea

What does ownership mean:
• Where is item stored?
• Where can it be displayed?
• Who receives royalties on item: owner or creator?
NFTs and DeFi: asset-based DeFi:

- Use NFT as collateral in loans (e.g., nftfi.com)
- Fractional ownership of NFT assets (e.g., fractional.art)
- NFT-based futures market

... all require a way to appraise an NFT (e.g., upshot.io)
Many more topics to cover
(1) Maximal extractable value (MEV):

• Recall: Ethereum v1 → all Tx enter a public mempool

• Example MEV problem:

(i) Trader Bob finds a liquidation opportunity on Compound,
(ii) Alice scans mempool, finds Bob’s Tx,
(iii) Alice issues Tx’ with higher gasPrice, scheduled first, and takes Bob’s profit

automated fontrunners ⇒ do this automatically
(1) **Maximal extractable value (MEV):**

- Recall: Ethereum v1 → all Tx enter a **public** mempool
- Example MEV problem:

Miner’s revenues increase (MEV). Who gets hurt?
- Bob. Leads to high gas prices on Ethereum, and other bad effects

What to do? Several answers: see, e.g., **flashbots** (mev-geth)
Many more topics to cover ... 

(1) Maximal extractable value (MEV)

(2) On-chain Governance:
   • How to decide on updates to Uniswap, Compound, ... ???
   • Current method:
     • Interested parties can buy governance tokens
     • One token one vote
   • Better mechanisms?
Example: Uniswap proposals

Add 1 Basis Point Fee Tier  executed
TLDR: Uniswap should add a 1bps fee tier with 1 tick spacing. This change is straightforward from a

Upgrade Governance Contract to Compound's Governor Bravo  executed
Previous Discussion: [Temperature Check](https://gov.uniswap.org/t/temperature-check-upgrade-gove...

Community-Enabled Analytics  canceled
*Past discussion:* [Temperature Check](https://gov.uniswap.org/t/temperature-check-larger-grant-pro

DeFi Education Fund  executed
##### (Previously known as: DeFi Political Defense Fund) Past discussion: [Temperature Check ](http

Reduce the UNI proposal submission threshold to 2.5M  executed
This proposal lowers the UNI proposal submission threshold from 10M UNI to 2.5M UNI. Uniswap's gove
Many more topics to cover ...

(1) Maximal extractable value (MEV)

(2) Project governance:
   • How to decide on updates to Uniswap, Compound, ... ???

(3) Insurance: against bugs in Dapp code and other hacks

(4) Many more cute cryptography techniques (see slides at end)

(5) Interoperability between blockchains ... discussed next
More topics ...

- Where can I learn more?
  - **CS255** and **CS355**: Cryptography
  - **EE374**: Scaling blockchains with fast consensus
  - Stanford blockchain club

Discussion: a career in blockchains? Where to start?
Bridging blockchains
Many L1 blockchains

**Bitcoin:** Bitcoin scripting language (with Taproot)

**Ethereum:** EVM. Currently: expensive Tx fees (better in Eth2)

EVM compatible blockchains: **Celo, Avalanche, BSC, ...**

- Higher Tx rate $\implies$ lower Tx fees
- EVM compatibility $\implies$ easy project migration and user support

Other fast non-EVM blockchains: **Solana, Flow, Algorand, ...**

- Higher Tx rate $\implies$ lower Tx fees
The problem: siloes

Can I use Serum??

20 DOT
Interoperability:

- User owns funds or assets (NFTs) on one blockchain system
  Goal: enable user to move assets to another chain

Composability:

- Enable a DAPP on one chain to call a DAPP on another

Both are easy if the entire world used Ethereum
- In reality: many blockchain systems that need to interoperate
- The solution: bridges
A first example: BTC in Ethereum

How to move BTC to Ethereum??  Goal: enable BTC in DeFi.

⇒ need new ERC20 on Ethereum pegged to BTC
  (e.g., use it for providing liquidity in DeFi projects)

The solution: wrapped coins
• Asset X on one chain appear as wrapped-X on another chain
• For BTC: several solutions  (e.g., wBTC, tBTC)
wBTC and tBTC: a lock-and-mint bridge

Let's start with wBTC: moving 1 BTC to Ethereum

1 BTC (lock 1 BTC)

Alice

1 BTC

custodian's BTC address

1 BTC verified (signed)

custodian

ERC20 bridge contract

mint 1 wBTC credit Alice's address

1 wBTC to use in DeFi

Alice on Ethereum

(watch for deposits)
Alice wants her 1 BTC back

Moving 1 wBTC back to the Bitcoin network:

Alice

1 $\mathbb{BTC}$

(1 BTC unlocked)

$\mathbb{BTC}$

custodian’s BTC address

$\mathbb{BTC}$

Bitcoin Tx (signed)

deduct 1 wBTC from Alice

burn my 1 wBTC (signed)

Alice on Ethereum

$\mathbb{BTC}$

Bridge contract

(watch for burns)

custodian

BANK
Example BTC → Ethereum:

- **Bitcoin Tx:** ≈4,000 BTC
- **Ethereum Tx:**

Why two hours? ... make sure no Bitcoin re-org

The problem: trusted custodian

Can we do better?
Alice requests to mint tBTC:
  random three registered custodians are selected and
  they generate P2PKH Bitcoin address for Alice
  signing key is 3-out-of-3 secret shared among three
  (all three must cooperate to sign a Tx)
  Alice sends BTC to P2PKH address, and received tBTC.

Custodians must lock 1.5x ETH stake for the BTC they manage
• If locked BTC is lost, Alice can claim staked ETH on Ethereum.
Bridging smart chains (with Dapp support)

A very active area:

- Many super interesting ideas

```
<table>
<thead>
<tr>
<th>Asset-specific</th>
<th>Chain-specific</th>
<th>Application-specific</th>
<th>Generalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>ever (AR)</td>
<td>Avalanche</td>
<td>ANY SWAP</td>
<td>AXELAR</td>
</tr>
<tr>
<td>INTERLAY (BTC)</td>
<td>BINANCE</td>
<td>Biconomy</td>
<td>Chainlink</td>
</tr>
<tr>
<td>CBTC</td>
<td>GRAVITY</td>
<td>CELER</td>
<td>ChainSafe</td>
</tr>
<tr>
<td>WBTC</td>
<td>Harmony</td>
<td>CHAINFLIP</td>
<td>composable</td>
</tr>
<tr>
<td>Rainbow Bridge</td>
<td>(PoS Bridge)</td>
<td>Gateway</td>
<td>connectx</td>
</tr>
<tr>
<td>Ronin</td>
<td>Rainbow Bridge</td>
<td>liqquality</td>
<td>deBridge</td>
</tr>
<tr>
<td>secret network</td>
<td>(SnowBridge)</td>
<td>Qredo</td>
<td>Ibc</td>
</tr>
<tr>
<td>Terra Shuttle</td>
<td>TokenBridge</td>
<td>Ren</td>
<td>LAYERZERO</td>
</tr>
<tr>
<td>TokenBridge</td>
<td>WORMHOLE</td>
<td>Synapse</td>
<td>Movr</td>
</tr>
<tr>
<td>WRAPPED</td>
<td>WRAP</td>
<td>THORCHAIN</td>
<td>OPTICS</td>
</tr>
<tr>
<td></td>
<td>(XCMC)</td>
<td>wanchain</td>
<td>Polymer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PolyNetwork</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ORBIT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Router</td>
</tr>
</tbody>
</table>

https://medium.com/1kxnetwork/blockchain-bridges-5db6afac44f8
```
Two types of bridges

Type 1: a lock-and-mint bridge
• SRC → DEST: user locks funds on SRC side, wrapped tokens are minted on the DEST side
• DEST → SRC: funds are burned on the DEST side, and released from lock on the SRC Side

Type 2: a liquidity pool bridge
• Liquidity providers provide liquidity on both sides
• SRC → DEST: user sends funds on SRC side, equivalent amount released from pool on DEST side
Bridging smart chains (with Dapp support)

Step 1 (hard): a secure cross-chain messaging system

Source Chain S

DAPP-X

message to Y on chain T: data

I believe it

Target Chain T

DAPP-Y

message from X on chain S: data

relayer (contract)

Step 2 (easier): build a bridge using messaging system
Bridging smart chains (with Dapp support)

**Step 1** (hard): a secure cross-chain messaging system

Source Chain S  DAPP-X  Target Chain T

**Step 2** (easier): build a bridge using messaging system

- DAPP-X → DAPP-Y: “I received 3 CELO, ok to mint 3 wCELO”
- DAPP-Y → DAPP-X: “I burned 3 wCELO, ok to release 3 CELO”

If messaging system is secure, no one can steal locked funds at S
Primarily two types of messaging systems

(1) **Externally verified**: external parties verify message on chain S

Relayer on S received messages D[] (signed)

Trustees (watch relayerS)

RelayerT dispatches only if all trustees signed

⇒ **if** DAPP-Y trusts trustees, it knows DAPP-X sent message
Primarily two types of messaging systems

(1) **Externally verified**: external parties verify message on chain S

- **Source Chain S**: relayerS
- **Target Chain T**: relayerT

Trustees (watch relayerS)

What if trustees sign and post a fake message to relayerT?

- off-chain party can send trustee’s signature to relayerS ⇒ trustee slashed
(2) **On-chain verified**: chain T verifies block header of chain S

---

**Source**
Chain S

**relayerS**

receive msgs

---

**Target**
Chain T

**relayerT**

verify and dispatch

send messages \(D[]\) to relayerT, along with **finalized** block header on chain S, and Merkle proofs

---

**oracle**

---

**no trustees**

**relayerT** runs a (light) client for chain S to verify that relayerS received messages \(D[]\)
Primarily two types of messaging systems

Source Chain S

relayerS

receive msgs

msgs D[]

SNARK prover

D[], BH, SNARK oracle

block header (BH) and Merkle proofs

Target Chain T

verify SNARK proof and dispatch

Problem: high gas costs on chain T to verify state of source chain.
Solution: use SNARKs $\implies$ little work for relayerT
Bridging: the future vision

User can hold assets on any chain

• Assets move cheaply and quickly from chain to chain

• A project’s liquidity is available on all chains

• Users and projects choose the chain that is best suited for their application and asset type

We are not there yet ...
Fun crypto tricks
Signatures make up most of Tx data.

Can we compress signatures?

- Yes: aggregation!
- not possible for ECDSA
BLS Signatures

Used in modern blockchains: Ethereum 2.0, Dfinity, Chia, etc.

The setup:

• $G = \{1, g, \ldots, g^{q-1}\}$ a cyclic group of prime order $q$

• $H: M \times G \rightarrow G$ a hash function (e.g., based on SHA256)
BLS Signatures

\textbf{KeyGen}(): choose random \( \alpha \) in \( \{1, \ldots, q\} \)
output \( \text{sk} = \alpha \), \( \text{pk} = g^\alpha \in G \)

\textbf{Sign}(sk, m): output \( \text{sig} = H(m, \text{pk})^\alpha \in G \)

\textbf{Verify}(pk, m, sig): output accept if \( \log_g(pk) = \log_{H(m,pk)}(sig) \)

Note: signature on \( m \) is unique! (no malleability)
How does verify work?

A pairing: an efficiently computable function $e: G \times G \rightarrow G'$ such that $e(g^\alpha, g^\beta) = e(g, g)^{\alpha\beta}$ for all $\alpha, \beta \in \{1, \ldots, q\}$ and is not degenerate: $e(g, g) \neq 1$

Observe: $\log_g(pk) = \log_{H(m,pk)}(\text{sig})$

if and only if $e(g, \text{sig}) = e(pk, H(m,pk))$

$e(g, H(m,pk)^\alpha) = e(g^\alpha, H(m,pk))$
Anyone can compress $n$ signatures into one

$pk_1, m_1 \rightarrow \sigma_1$

$\vdots$

$pk_n, m_n \rightarrow \sigma_n$

 aggregate $\rightarrow \sigma^*$

Verify($pk, \overline{m}, \sigma^*$) = “accept”

convinces verifier that

for $i=1,...,n$:  
user $i$ signed msg $m_i$
Verifying an aggregate signature: (incomplete)

\[
\prod_{i=1}^{n} e(H(m_i, p_{k_i}), g^{\alpha_i}) \stackrel{?}{=} e(\sigma, g)
\]

\[
\prod_{i=1}^{n} e(H(m_i, p_{k_i})^{\alpha_i}, g) = e(\prod_{i=1}^{n} H(m_i, p_{k_i})^{\alpha_i}, g)
\]
Compressing the blockchain with BLS

If needed:
- Compress all signatures in a block into a single aggregate signature

⇒ Shrink block

Or: Aggregate in smaller batches
Reducing Miner State
Miners need to keep all UTXOs in memory to validate Txs

Can we do better?
Recall: polynomial commitments

- $\text{commit}(pp, f, r) \rightarrow \text{com}_f$ commitment to $f \in \mathbb{F}_p^{(\leq d)}[X]$

- $\text{eval}$: goal: for a given $\text{com}_f$ and $x, y \in \mathbb{F}_p$,
  
  construct a SNARK to prove that $f(x) = y$. 

A polynomial commitment is **homomorphic** if there are efficient algorithms such that:

- \( \text{commit}(pp, f_1, r_1) \rightarrow \text{com}_{f_1} \quad \text{commit}(pp, f_2, r_2) \rightarrow \text{com}_{f_2} \)

Then:

(i) for all \( a, b \in \mathbb{F}_p \) : \( \text{com}_{f_1}, \text{com}_{f_2} \rightarrow \text{com}_{a*f_1+b*f_2} \)

(ii) \( \text{com}_{f_1} \rightarrow \text{com}_{x*f_1} \)
Committing to a set (of UTXOs)

Let $S = \{U_1, ..., U_n\} \in \mathbb{F}_p$ be a set of UTXOs.

Define: $f(X) = (X - U_1) \cdots (X - U_n) \in \mathbb{F}_p^{(\leq n)}[X]$.

Set: $\text{com}_f = \text{commit}(pp, f, r)$ $\leftarrow$ short commitment to $S$.

For $U \in \mathbb{F}_p$: $U \in S$ if and only if $f(U) = 0$.

To add $U$ to $S$: $\text{com}_f \rightarrow \text{com}_{X^*f - U^*f}$ $\leftarrow$ short commitment to $S \cup \{U\}$.
Miners maintain two commitments:

(i) commitment to set $T$ of all UTXOs
(ii) commitment to set $S$ of spent TXOs

\[ \{ \text{com}_T, \text{com}_S \} \leq 1\text{KB} \]

**Tx format:**
- every input $U$ includes a proof \((U \in T \land U \notin S)\)
  Two eval proofs: \(T(U) = 0 \land S(U) \neq 0\) (short)

**Tx processing:** miners check eval proofs, and if valid, add inputs to set $S$ and outputs to set $T$. That’s it!
Problem: how does a user prove that her UTXO $U$ satisfies

$$T(U) = 0 \quad \&\& \quad S(U) \neq 0$$

This requires knowledge of the entire blockchain

$\Rightarrow$ user needs large memory and compute time

$\Rightarrow$ ... can be outsourced to an untrusted 3rd party
Is this practical?

Not quite ...

- Problem: the factory’s work per proof is **linear** in the number of UTXOs ever created

- **Many** variations on this design:
  - can reduce factory’s work to $\log_2(\# \text{ current UTXOs})$ per proof
  - Factory’s memory is linear in $(\# \text{ current UTXOs})$

End result: outsource memory requirements to a small number of 3rd party service providers
Taproot: semi-private scripts in Bitcoin
Bitcoin's long-anticipated Taproot upgrade is activated
Currently: Bitcoin scripts must be fully revealed in spending Tx

Can we keep the script secret?

Answer: Yes, easily! when all goes well ...
ECDSA and Schnorr public keys:

- **KeyGen()**: $sk = \alpha$, $pk = g^\alpha \in G$ for $\alpha$ in $\{1, \ldots, q\}$

Suppose $sk_A = \alpha$, $sk_B = \beta$.

- Alice and Bob can sign with respect to $pk = pk_A \cdot pk_B = g^{\alpha+\beta}$

  $\Rightarrow$ an interactive protocol between Alice and Bob

  (note: much simpler with BLS)

  $\Rightarrow$ Alice & Bob can imply consent to Tx by signing with $pk = g^{\alpha+\beta}$
S: Bitcoin script that must be satisfied to spend a UTXO $U$

S involves only Alice and Bob. Let $pk_{AB} = pk_A \cdot pk_B$

Goal: keep S secret when possible.

How: modify S so that a signature with respect to

$$pk = pk_{AB} \cdot g^{H(pk_{AB}, S)}$$

is sufficient to spend UTXO, without revealing S!!
The main point

- If parties agree to spend UTXO,
  ⇒ sign with respect to $pk_{AB}$ and spend while keeping $S$ secret

- If disagreement, Alice can reveal $S$
  and spend UTXO by proving that she can satisfy $S$.

Taproot pk compactly supports both ways to spend the UTXO
Next lecture: super cool final guest lecture