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

rollup server

Merkle Tree

[L1 blockchain (e.g. Ethereum)]

block 354

atomic swap:

[A→B: 2 ETH], $\text{sig}_A$

[B→Z: 1 ETH]

[Z→B: 2 BAT] $\text{sig}_B$ $\text{sig}_Z$

Tx

Alice: 5 DAI
3 ETH

Bob: 2 ETH

Zoe: 1 ETH
3 BAT
Quick Recap: zkRollup

rollup server

new root

Merkle Tree

atomic swap:

\[[A \rightarrow B: 2 \text{ ETH}], \, \text{sig}_A\]

\[[B \rightarrow Z: 1 \text{ ETH}]\]
\n\[[Z \rightarrow B: 2 \text{ BAT}]
\\text{sig}_B \, \text{sig}_Z\]

Alice:
5 DAI
1 ETH

Bob:
3 ETH
2 BAT

... 

Zoe:
2 ETH
1 BAT

Tx data, SNARK

L1 blockchain
(verifies SNARK)

block 354

block 357
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>Offered</th>
<th>Sold</th>
<th>Claimed</th>
</tr>
</thead>
<tbody>
<tr>
<td>bid beautiful</td>
<td>visa</td>
<td>7610</td>
<td></td>
</tr>
<tr>
<td>sold gmoney</td>
<td>7610</td>
<td>0xa04e64</td>
<td>7610</td>
</tr>
<tr>
<td>bid 0xa04e64</td>
<td>0xa04e64</td>
<td>0xa04e64</td>
<td>0xa04e64</td>
</tr>
<tr>
<td>sold gr8wxl</td>
<td>0x84c920</td>
<td>21Ω ($31,117)</td>
<td>0x84c920</td>
</tr>
<tr>
<td>offered</td>
<td>21Ω ($31,117)</td>
<td>0x02751f</td>
<td>21Ω ($31,117)</td>
</tr>
<tr>
<td>sold 0x02751f</td>
<td>0x02751f</td>
<td>0.30Ω ($67)</td>
<td>0x02751f</td>
</tr>
<tr>
<td>offered</td>
<td>0.30Ω ($67)</td>
<td>0.30Ω ($59)</td>
<td></td>
</tr>
<tr>
<td>claimed</td>
<td></td>
<td></td>
<td>0.30Ω ($59)</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/
digital assets: where is this going?

NFTs are about managing ownership of general digital assets.

Growing list of categories on OpenSea:
- Art
- Music
- Domain Names
- Virtual Worlds
- Trading Cards
- Collectibles
- Sports
- Utility

What does ownership mean:
- Where is item stored?
- Where can it be displayed?
- Who receives royalties on item: owner or creator?
digital assets: where is this going?

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 $\implies$ 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 $\implies$ do this automatically
(1) Maximal extractable value (MEV):

- Recall: Ethereum v1 \(\implies\) 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??

How???

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

Alice on Ethereum

1 BTC

(lock 1 BTC)

(.watch for deposits)

1 BTC verified (signed)

bridge contract

mint 1 wBTC to use in DeFi

1 wBTC

ERC20

credit Alice’s address

BANK

custodian

custodian’s BTC address

Alice

1 BTC

(watch for deposits)
Alice wants her 1 BTC back

Moving 1 wBTC back to the Bitcoin network:

- Alice on Ethereum
- burn my 1 wBTC (signed)
- custodian’s BTC address
- deduct 1 wBTC from Alice
- Bitcoin Tx (signed)
- bridge contract
- (watch for burns)
- custodian
- 1 BTC unlocked
**wBTC**

Example BTC → Ethereum:

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov 26 2021 - 07:36</td>
<td>FUNDS SENT TO CUSTODIAN</td>
<td>c605b4f2f0948e7deae0c5d7c27b3256b97120be760e2b81136eb95c819570f6</td>
</tr>
<tr>
<td>Nov 26 2021 - 09:50</td>
<td>MINT COMPLETED BY CUSTODIAN</td>
<td>0x70475eca8be89b67143f1b52df013fc1df7d254e836c836c8f368fc516aca76b</td>
</tr>
</tbody>
</table>

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

**CUSTODY** Nov. 2021

- **253,387,2485 BTC** ($14,268,319,582.44 USD)

The problem: trusted custodian

Can we do better?
tBTC: no single point of trust

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><strong>ever</strong>&lt;sup&gt;(AR)&lt;/sup&gt;</td>
<td><strong>Avalanche</strong></td>
<td><strong>ANY SWAP</strong>&lt;sup&gt;*&lt;/sup&gt;</td>
<td><strong>AXELAR</strong>&lt;sup&gt;*&lt;/sup&gt;</td>
</tr>
<tr>
<td><strong>INTERLAY</strong>&lt;sup&gt;(BTC)&lt;/sup&gt;</td>
<td><strong>BINANCE</strong></td>
<td><strong>Biconomy</strong></td>
<td><strong>Chainlink</strong></td>
</tr>
<tr>
<td><strong>GBTC</strong></td>
<td><strong>GRAVITY</strong></td>
<td><strong>CELER</strong></td>
<td><strong>ChainSafe</strong></td>
</tr>
<tr>
<td><strong>WBTC</strong></td>
<td><strong>Harmony</strong></td>
<td><strong>CHAINFLIP</strong></td>
<td><strong>composable</strong></td>
</tr>
<tr>
<td><strong>WRAPPED</strong></td>
<td><strong>(PoS Bridge)</strong></td>
<td><strong>Gateway</strong></td>
<td><strong>connex</strong></td>
</tr>
<tr>
<td><strong>Rainbow Bridge</strong></td>
<td><strong>Ronin</strong></td>
<td><strong>liquidity</strong></td>
<td><strong>deBridge</strong></td>
</tr>
<tr>
<td><strong>SnowBridge</strong></td>
<td><strong>secret network</strong>&lt;sup&gt;®&lt;/sup&gt;</td>
<td><strong>Qredo</strong></td>
<td><strong>LayerZero</strong></td>
</tr>
<tr>
<td><strong>Terra Shuttle</strong></td>
<td><strong>TokenBridge</strong></td>
<td><strong>Ren</strong></td>
<td><strong>Movr</strong></td>
</tr>
<tr>
<td><strong>WRAP</strong></td>
<td><strong>wanchain</strong></td>
<td><strong>Synapse</strong></td>
<td><strong>OPTICS</strong></td>
</tr>
<tr>
<td><strong>WRAP</strong>&lt;sup&gt;(XCMPl)&lt;/sup&gt;</td>
<td></td>
<td></td>
<td><strong>PolyNetwork</strong></td>
</tr>
</tbody>
</table>

https://medium.com/1kxnetwork/blockchain-bridges-5db6afac44f8
Bridging smart chains (with Dapp support)

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

- **Source Chain S**: DAPP-X
- **Target Chain T**: DAPP-Y

**DAPP-X** sends a message to **DAPP-Y** via a relayer (contract).

**I believe it**

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

- **Source Chain S**: DAPP-X
- **Target Chain T**: DAPP-Y

**DAPP-X** sends a message to **DAPP-Y** via a relayer (contract).
Bridging smart chains (with Dapp support)

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

- Source Chain S
- DAPP-X
- Target Chain T
- DAPP-Y

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

Relayer on S received messages \(D[\cdot]\) (signed)

RelayerT dispatches only if all trustees signed

\[\Rightarrow \text{if DAPP-Y trusts trustees, it knows DAPP-X sent message}\]
Primarily two types of messaging systems

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

- **Source Chain S**: relayerS
  - collect msgs D[]
- **Target Chain T**: relayerT
  - verify sig and dispatch to recipients
- Trustees (watch relayerS)

Relayer on S received messages D[] (signed)

What if trustees sign and post a fake message to relayerT?
- off-chain party can send trustee’s signature to relayerS ⇒ trustee slashed
Primarily two types of messaging systems

(2) **On-chain verified**: chain T verifies block header of chain S

- **Source** Chain S
  - **relayerS**
    - receive msgs
    - send messages D[] to relayerT, along with **finalized** block header on chain S, and Merkle proofs
    - oracle

- **Target** Chain T
  - **relayerT**
    - verify and dispatch
    - **no trustees**

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

Problem: high gas costs on chain T to verify state of source chain.
Solution: use SNARKs ⇒ 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

**KeyGen()**: choose random \( \alpha \) in \( \{1, \ldots, q\} \)

output \( sk = \alpha, \quad pk = g^\alpha \in G \)

**Sign**(sk, \( m \)): output \( sig = H(m, pk)^\alpha \in G \)

**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 $\frac{e(g, \text{sig})}{e(pk, H(m, pk))} = \frac{e(g^\alpha, H(m, pk))}{e(g, H(m, pk)^\alpha)}$
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($\overline{pk}, \overline{m}, \sigma^*$) = “accept”

convinces verifier that for $i=1,\ldots,n$:
user $i$ signed msg $m_i$
Aggregation: how

\begin{align*}
\text{user 1: } \quad & pk_1 = g^{\alpha_1}, \quad m_1 \rightarrow \quad \sigma_1 = H(m_1, pk_1)^{\alpha_1} \\
\vdots \quad & \\
\text{user n: } \quad & pk_n = g^{\alpha_n}, \quad m_n \rightarrow \quad \sigma_n = H(m_n, pk_n)^{\alpha_n}
\end{align*}

\[\sigma \leftarrow \sigma_1 \cdots \sigma_n\]

Verifying an aggregate signature: (incomplete)

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

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

one Bitcoin block

if needed:
compress all signatures in a block into a single aggregate signatures

⇒ 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

• $\text{commit}(pp, f, r) \to \text{com}_f$ 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 $f(x) = y$. 
Homomorphic polynomial commitment

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

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

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

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

To add $U$ to $S$: $\text{com}_f \rightarrow \text{com}_{X^*f-U^*f}$ ← short commitment to $S \cup \{U\}$
How does this help?

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!
Does this work??

**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

---

**Diagram:**

- **UTXO $U$, fee** input
- **polynomials S and T** output
- **proof $\pi$** input
- **The proof factory**
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 (\# current UTXOs)

End result: outsource memory requirements to a small number of 3rd party service providers
Taproot: semi-private scripts in Bitcoin
Taproot is here ...

Bitcoin's long-anticipated Taproot upgrade is activated

November 14, 2021, 12:49AM EST · 1 min read
Script privacy

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 \( \{1, ..., q\} \)

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 \( pk = g^{\alpha+\beta} \)
How?

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,
  \[ \Rightarrow \text{ sign with respect to } p_k_{AB} \text{ and spend while keeping } S \text{ secret} \]

• 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
END OF LECTURE

Next lecture: super cool final guest lecture