CS251 Fall 2022

(cs251.stanford.edu)

Final Topics: Bridging and MEV

Dan Boneh

Invited talk final lecture.   Final exam next Wednesday.

… but first, final thoughts on ZK

Commercial interest in SNARKs

STARKWARE

Aztec

Matter Labs

ESPRESSO SYSTEMS

RISC Zero

Scroll

polygon

Aleo

SUPRA NATIONAL

INGONYAMA

Many more building applications on top …

Why so much commercial interest?

Babai-Fortnow-Levin-Szegedy 1991:

an L1 blockchain

In this setup, ~~a single reliable PC~~ can monitor the operation of a herd of ~~supercomputers~~ working with unreliable software.

coordinators

“Checking Computations in Polylogarithmic Time”

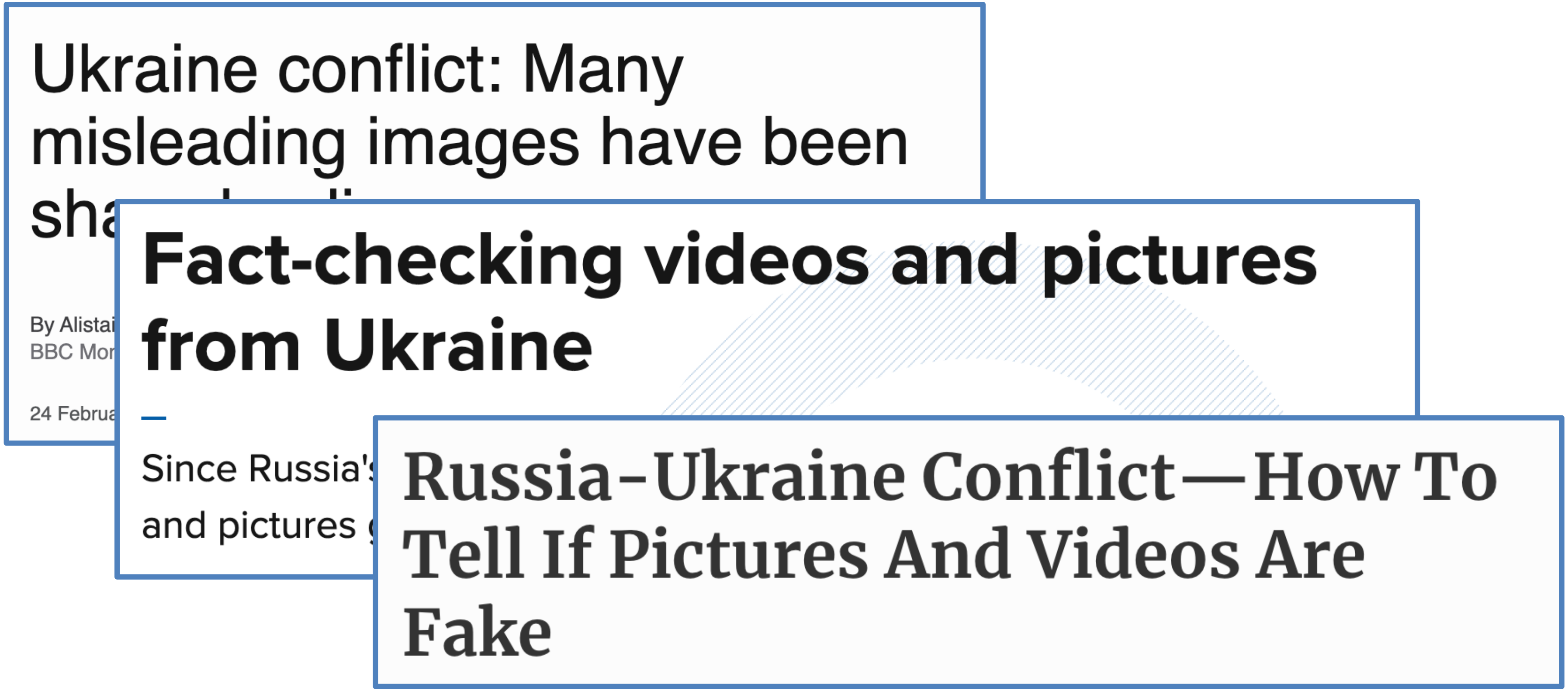
We are going to the moon …

Blockchains drive the development of SNARKs:

zkRollup,   zkBridge, zkCreditScore, zkProofOfSolvency, …

… but **<u>many</u>** non-blockchain applications

Using ZK to fight disinformation



Ukraine conflict: Many misleading images have been sha

By Alistai
BBC Mon

24 Februa

Fact-checking videos and pictures from Ukraine

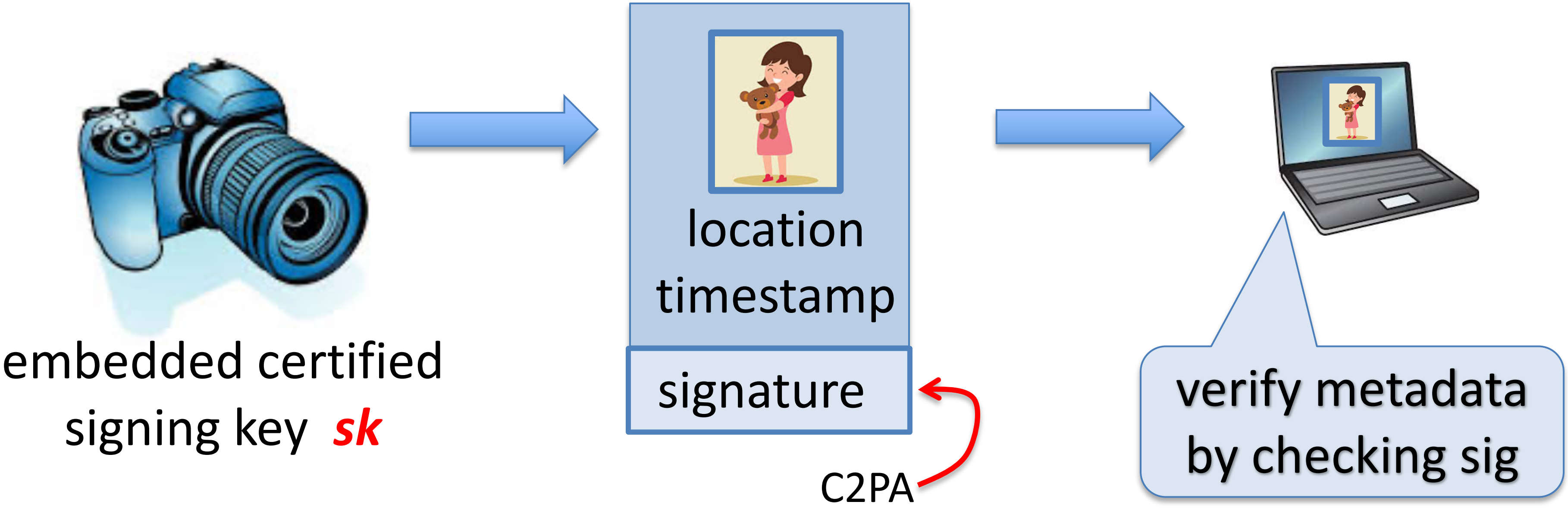
Since Russia's
and pictures

Russia-Ukraine Conflict—How To Tell If Pictures And Videos Are Fake

C2PA: a standard for content provenance

Sony Unlocks In-Camera Forgery-Proof Technology

04 Aug, 2022



embedded certified signing key ***sk***

location

timestamp

signature

C2PA

verify metadata by checking sig

A problem: post-processing

Newspapers often process the photos before publishing:

- Resize (1500 × 1000), Crop, Grayscale (AP lists allowed ops)

The problem: laptop cannot verify signature on processed photo

C2PA "solution":
editing software will sign
processed photo to certify edits

A solution using ZK proofs (SNARKs)

(with T. Datta)

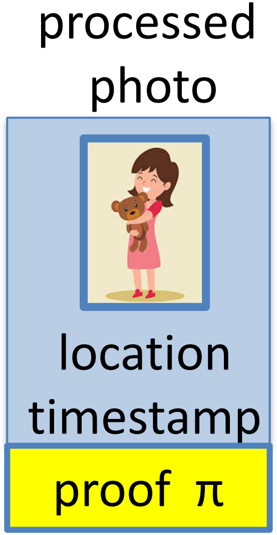
Editing software attaches a proof π to (processed) photo that:

I know a triple (***Orig***, ***Ops***, ***Sig***) such that

1. ***Sig*** is a valid C2PA signature on ***Orig***

2. photo is the result of applying ***Ops*** to ***Orig***

3. metadata(photo) = metadata(***Orig***)



processed photo

location

timestamp

proof π

$\Rightarrow$ Laptop verifies π and shows metadata to user

Performance

Proof size: 200-400 bytes. Verification time: 2 ms.
(in browser)

Proof generation time by newspaper:

- Resize (3000 × 3000 → 1500 × 1500): 84 sec.
- Crop (3000 × 3000 → 1500 × 1500): 60 sec.
- Grayscale (2.25M pixels): 25 sec.

What about video??

See also: PhotoProof by Naveh & Tromer (2016)

Many more topics …

Many more topics to cover …

(1) Maximal extractable value (MEV)

(2) Blockchain interoperability (bridging)

(3) Project governance: (see our Spring course on DAOs)

- How to decide on updates to Uniswap, Compound, … ???

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

(5) Many more cryptography techniques (see slides at end)

More topics …

- Where can I learn more?
  - **CS255** and **CS355**: Cryptography (Winter and Spring)
  - **EE374**: Scaling blockchains with fast consensus (Winter)
  - **Stanford blockchain conference** (SBC): Aug. 28-30, 2023.
  - **Stanford blockchain club**

A career in blockchains? Where to start? [link]

Maximal Extractable Value (MEV)

Searchers

Ethereum gives rise to a new type of business: **searchers**

- **Arbitrage:** Uniswap DAI/USDC exchange rate is 1.001
  whereas at Sushiswap the rate is 1.002

  ⇒ a searcher posts Tx to equalize the markets and profits

- **Liquidation**: suppose there is a liquidation opportunity on Aave

  ⇒ a searcher posts a liquidation Tx and profits

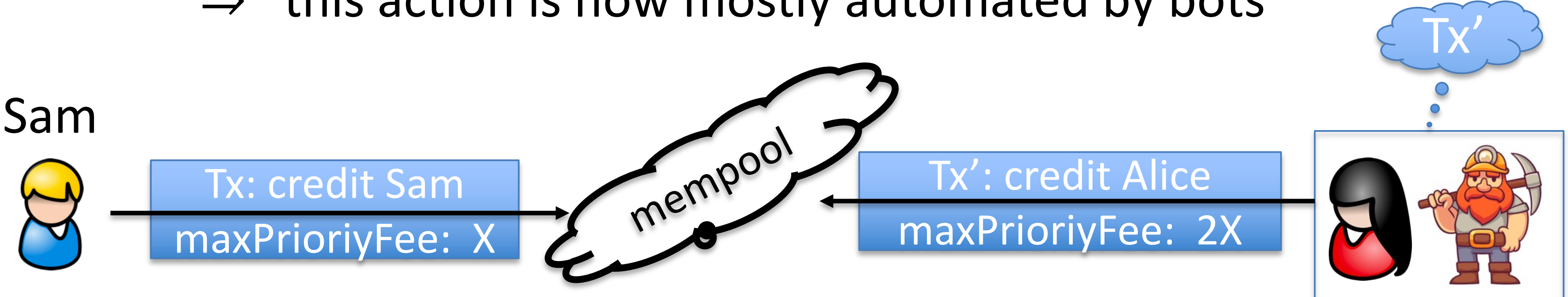
- Many other examples … often using a sequence of Tx (a bundle)

The MEV problem

What happens when a searcher posts a Tx to the mempool?

- **Validator:** create a new Tx’ with itself as beneficiary, and place it before Sam’s Tx in the proposed block
- **Another searcher:** create a new Tx’ with itself as beneficiary, and posts it with a higher *maxPrioriyFee*

⇒ this action is now mostly automated by bots

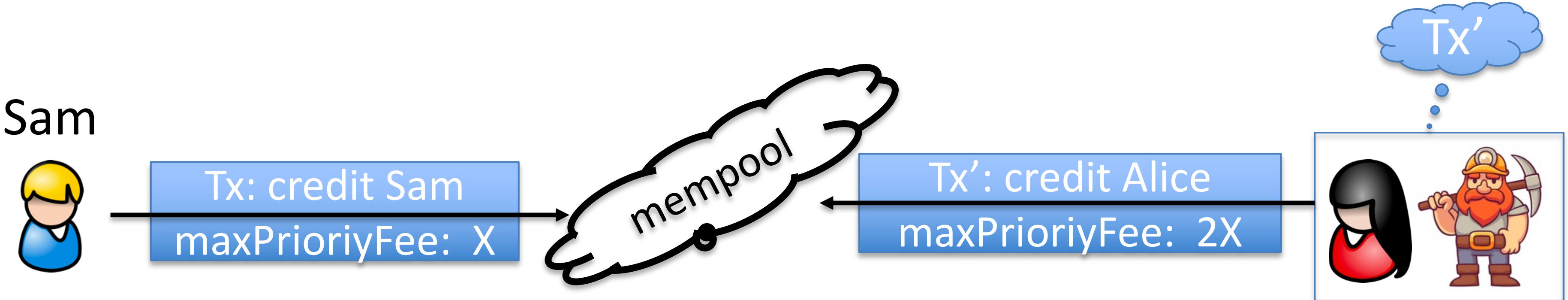


The result harms honest users

Price Gas Auctions (PGA): two or more searchers compete

- Repeatedly submit a Tx with higher and higher *maxPriorityFee* until a validator chooses one … happens within a few seconds

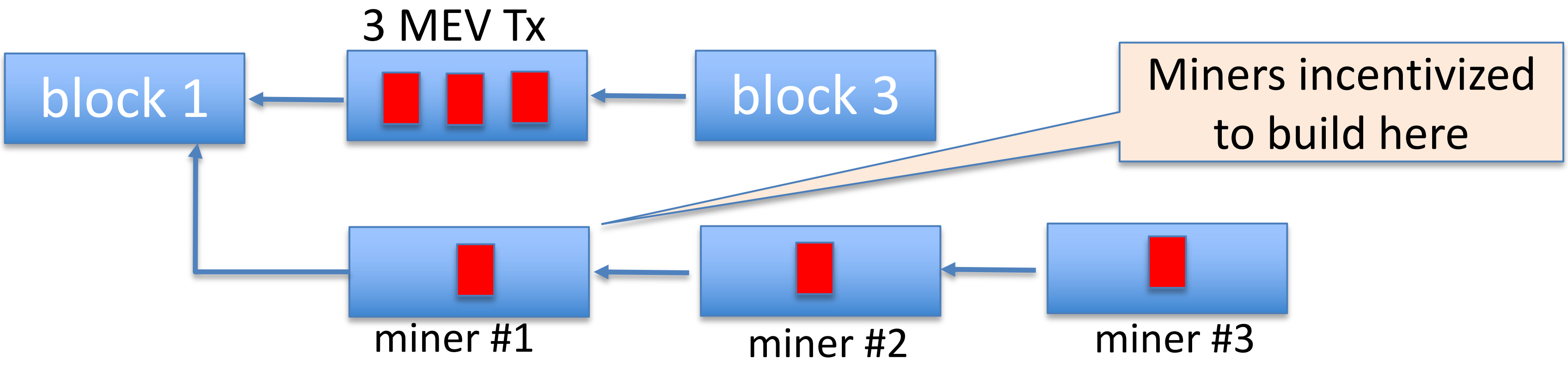
⇒ causes congestion (lots of Tx in mempool) and high gas fees



The result harms consensus

Undercutting attack on longest-chain consensus:

Rational miner: can cause a re-org by taking one MEV Tx for itself and leave two for other miners



The problem: MEV Tx generate extra revenue for miners, higher than block rewards

The result causes centralization

Validators can steal MEV Tx from searchers ⇒ **Private mempools**

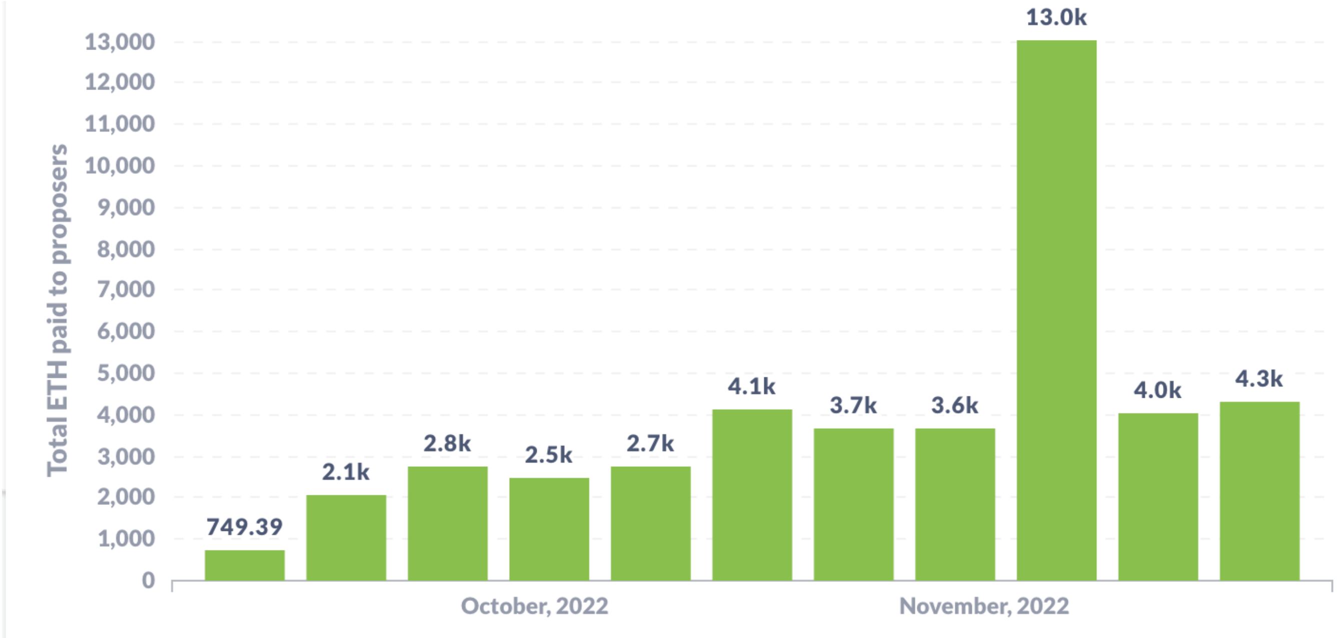
Searchers only send Tx to a validator they trust
(have a business relation with)

These validators do not propagate Tx to the network,
but put them in blocks themselves

In the long run: a few validators will handle the bulk of all Tx

How big are MEV rewards?

Weekly MEV amount paid to validators (in ETH):



source: transparency.flashbots.net

What to do??

Proposer Builder Separation (PBS)

Goals:

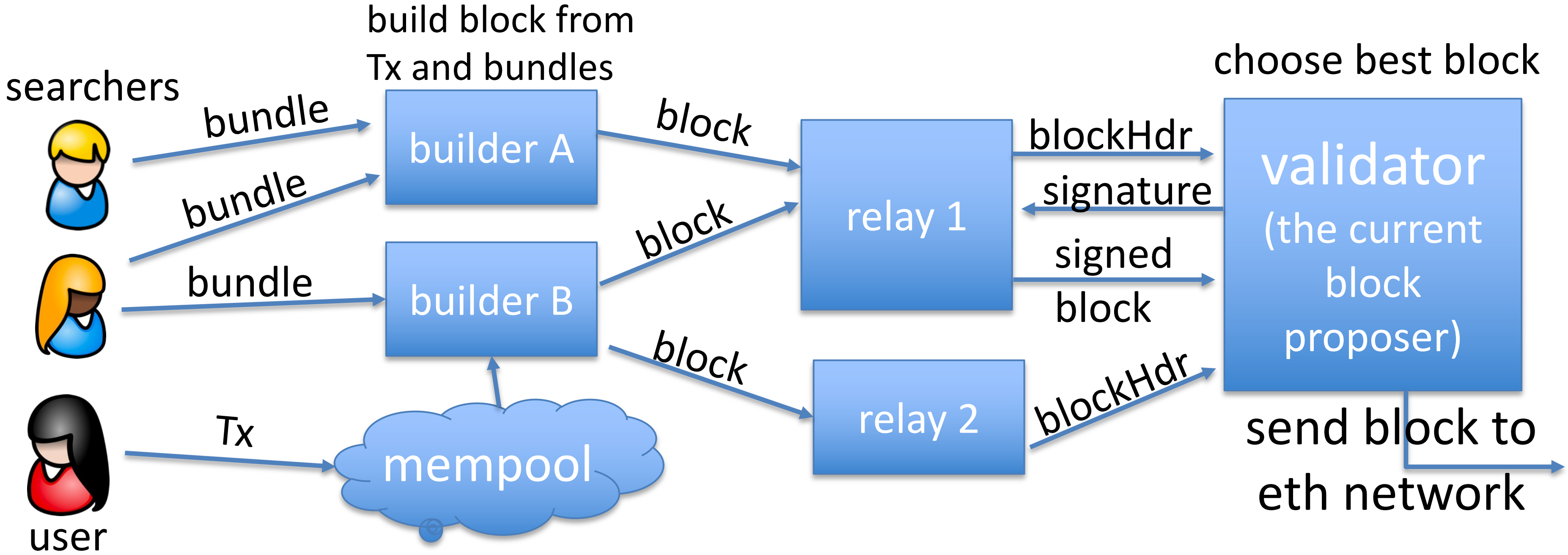
- Eliminate price gas auctions in the public mempool
  - Instead, create an open market for searchers to compete on the position of their bundles in a block
- Prevent validator concentration: make it possible for every validator to earn MEV payments from searchers

Current PBS implementation: **MEV-boost**

The participants in PBS (as in MEV-boost)

Users have Tx and searchers have bundles (sequence of Tx)

- searcher wants its bundle posted in a block unmodified



MEV-boost

Builder: collects bundles and Tx and builds a block

- includes a MEV offer to validator (feeRecipient)

Relay: collects blocks, chooses block with max MEV offer

- sends block header (and MEV offer) to block proposer
- Can’t expose Tx in block to proposer (or proposer could steal Tx)

Proposer: chooses best offer and signs header with its staking key

⇒ Then Relay reveals block contents; proposer sends to network

(if bad block, proposer can build a block locally from mempool)

https://writings.flashbots.net/searching-post-merge

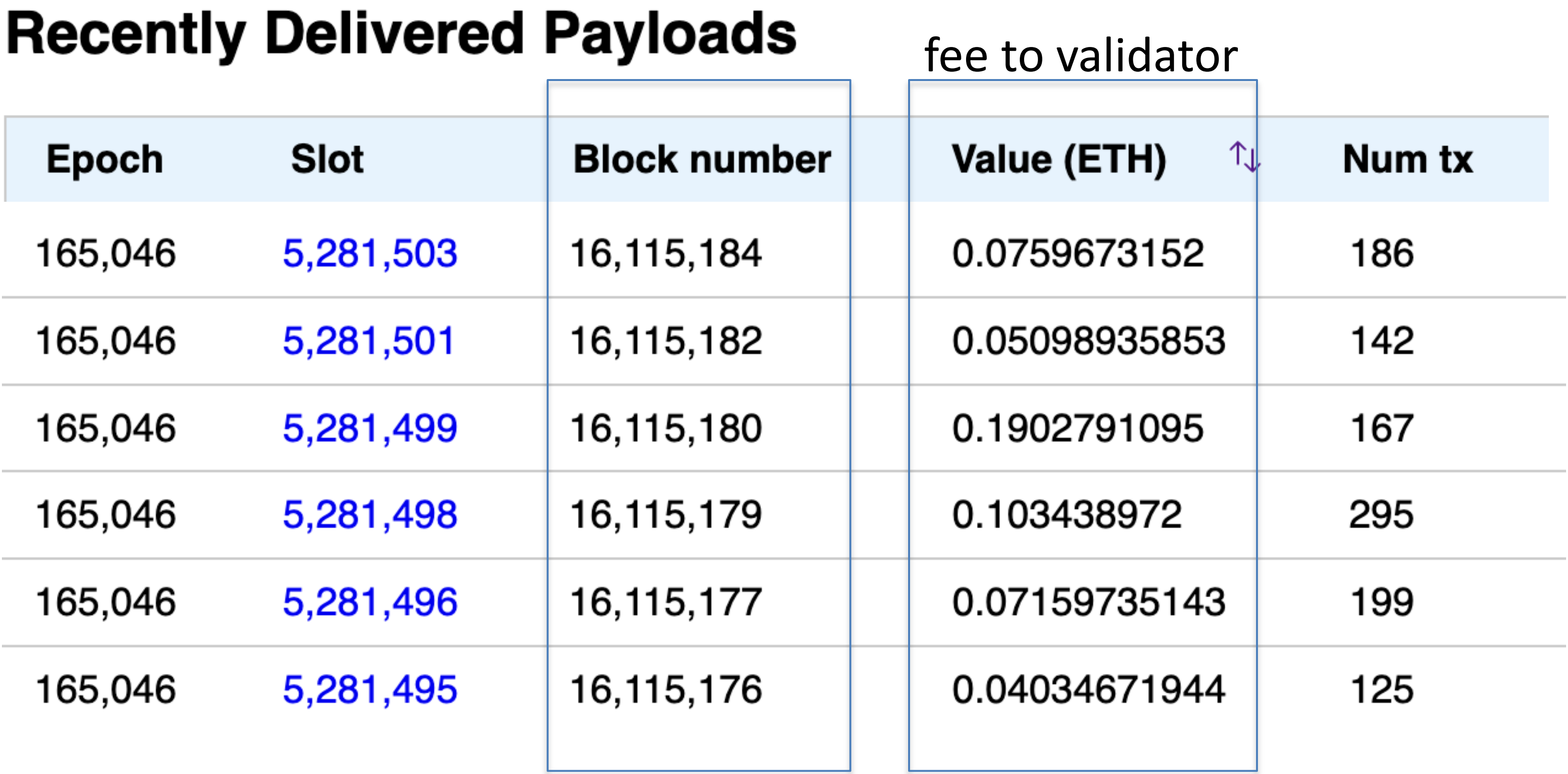
Operating relays

Flashbots: Filters out OFAC sanctioned addresses,
aims to maximize validator payout
(so that many validators will work with it)

BloXroute: no censorship
aims to maximize validator payout

…

An example: flashbots relay

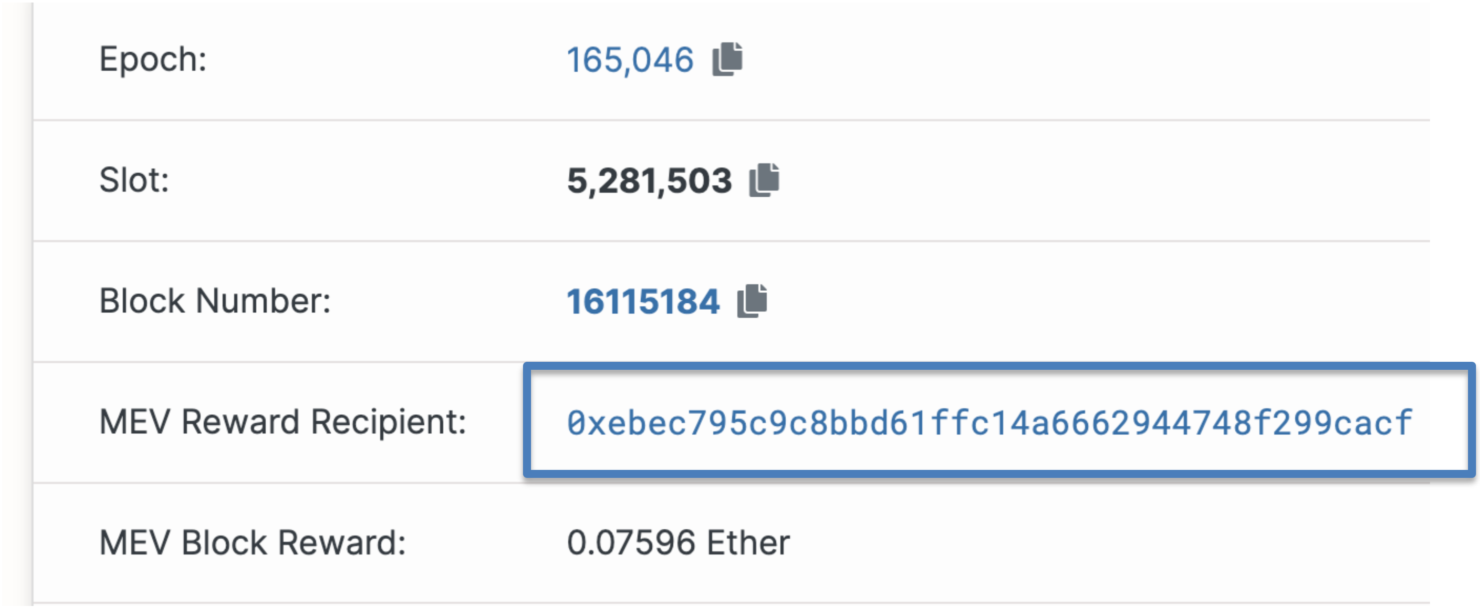


Recently Delivered Payloads

Epoch	Slot	Block number	Value (ETH) ↑↓	Num tx
165,046	5,281,503	16,115,184	0.0759673152	186
165,046	5,281,501	16,115,182	0.05098935853	142
165,046	5,281,499	16,115,180	0.1902791095	167
165,046	5,281,498	16,115,179	0.103438972	295
165,046	5,281,496	16,115,177	0.07159735143	199
165,046	5,281,495	16,115,176	0.04034671944	125

fee to validator

An example: flashbots relay



Epoch:	165,046
Slot:	**5,281,503**
Block Number:	**16115184**
MEV Reward Recipient:	0xebec795c9c8bbd61ffc14a6662944748f299cacf
MEV Block Reward:	0.07596 Ether

address of validator who proposed the block

Are we done?   Not quite …

Over the last 30 days: five block builders built 80% of all blocks !!

- Clear centralization in the builder market
- Enables censorship by builders

MEV-boost is not designed for cross-chain MEV

- For cross-chain arbitrage, no atomicity guarantee for bundle

A solution: SUAVE   (not yet deployed)

Interoperability between blockchain

How to bridge chains

Many L1 blockchains

Bitcoin: Bitcoin scripting language (with Taproot)

Ethereum: EVM. Currently: high Tx fees (better with Rollups)

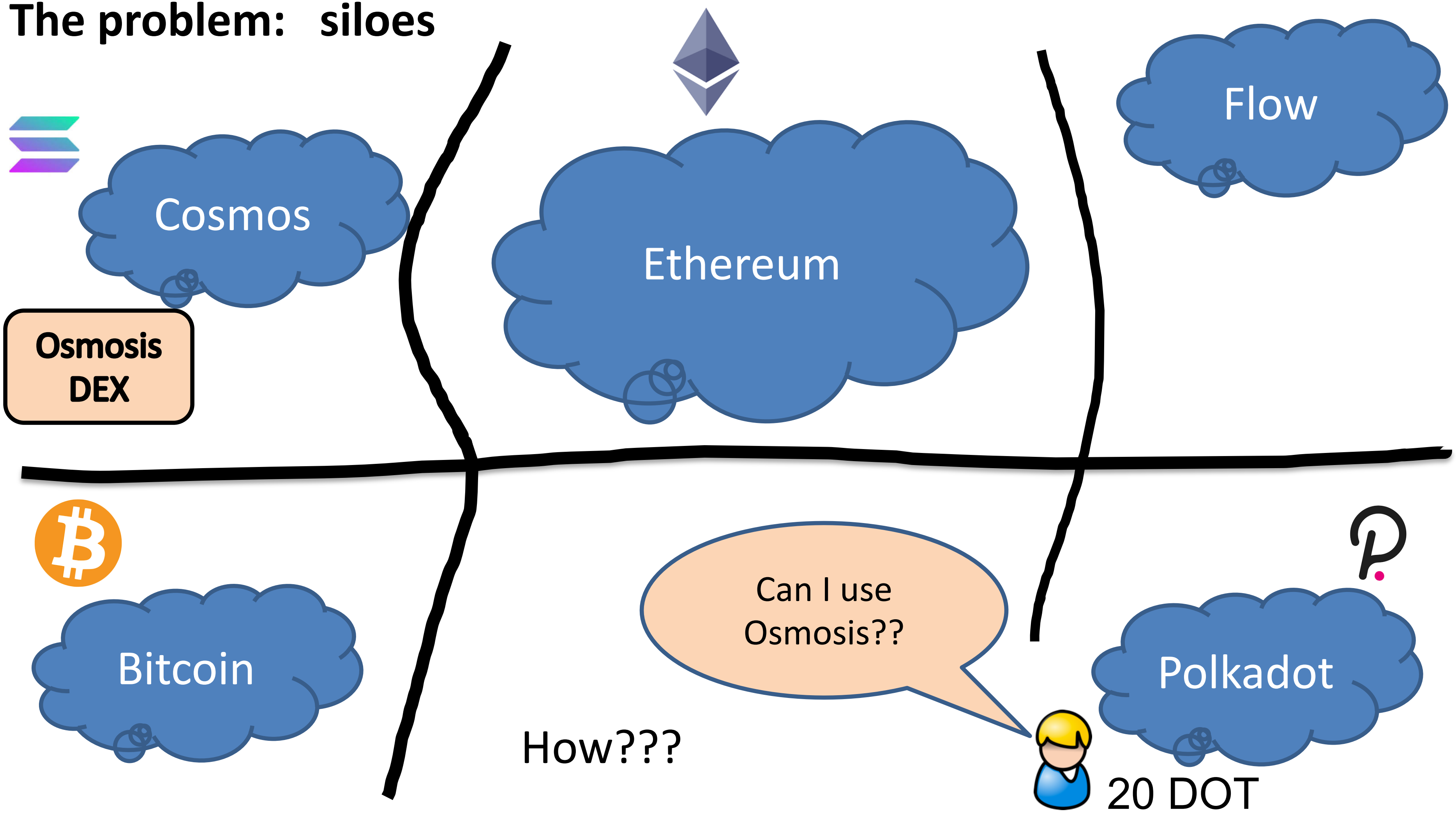
EVM compatible blockchains: **Celo, Avalanche, BSC**, …

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

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

- Higher Tx rate $\Longrightarrow$ lower Tx fees

The problem: siloes



Cosmos

Osmosis DEX

Ethereum

Flow

Bitcoin

Can I use Osmosis??

How???

Polkadot

20 DOT

Interoperability

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

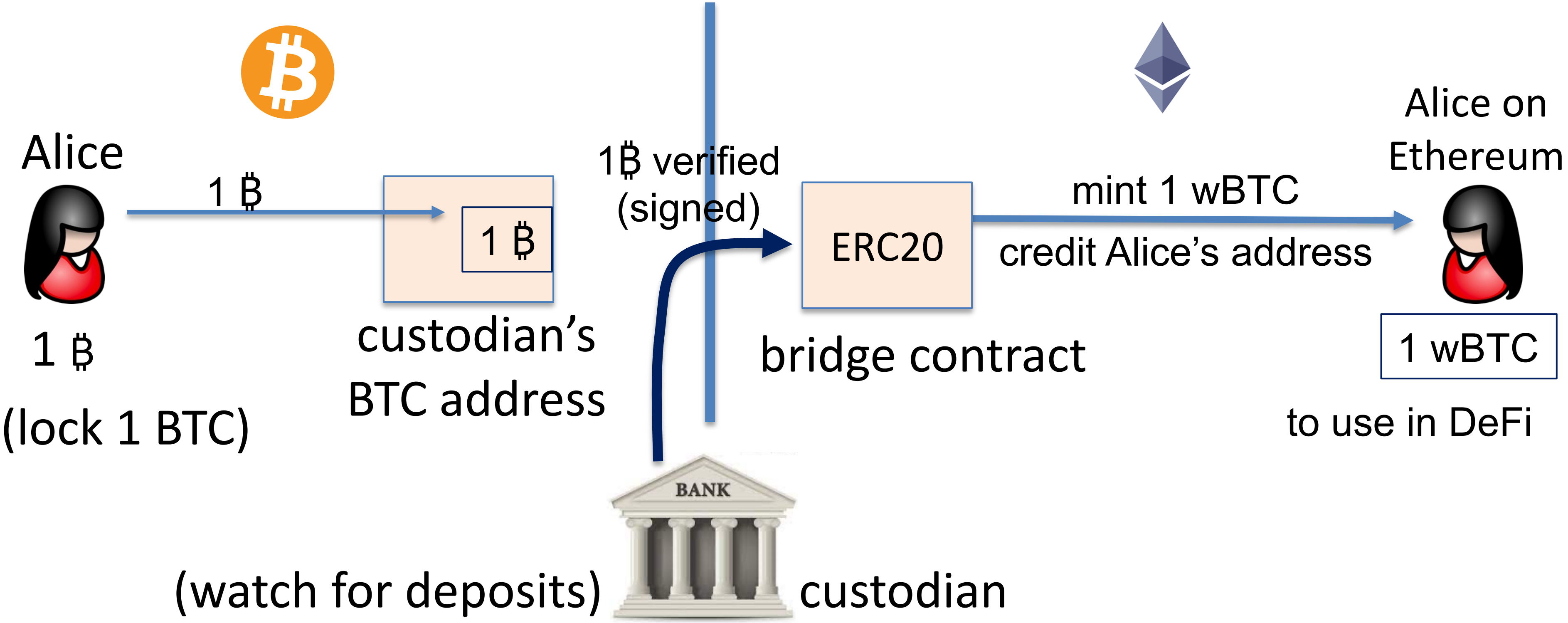
(e.g., use it for providing liquidity in DeFi projects)

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**



Alice

1 ฿

1 ฿

(lock 1 BTC)

1 ฿

custodian’s BTC address

1฿ verified (signed)

ERC20

bridge contract

mint 1 wBTC

credit Alice’s address

Alice on Ethereum

1 wBTC

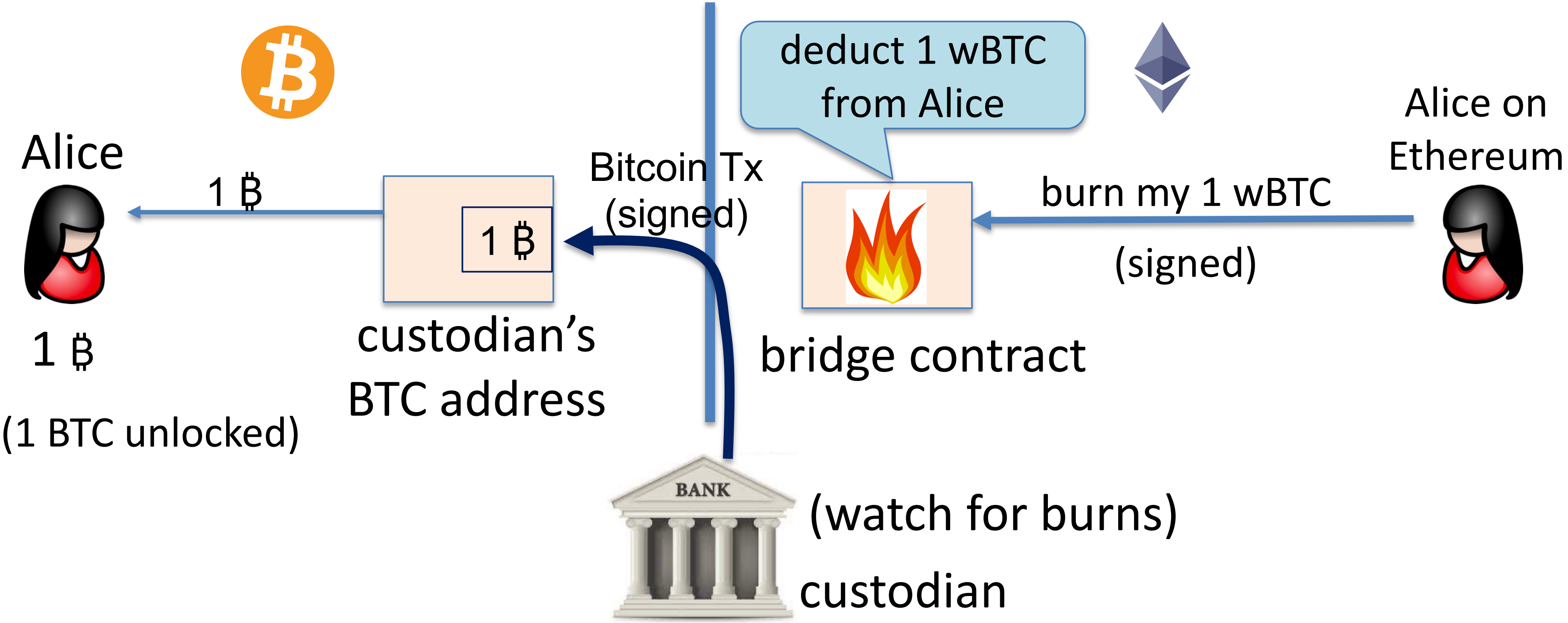
to use in DeFi

(watch for deposits)

custodian

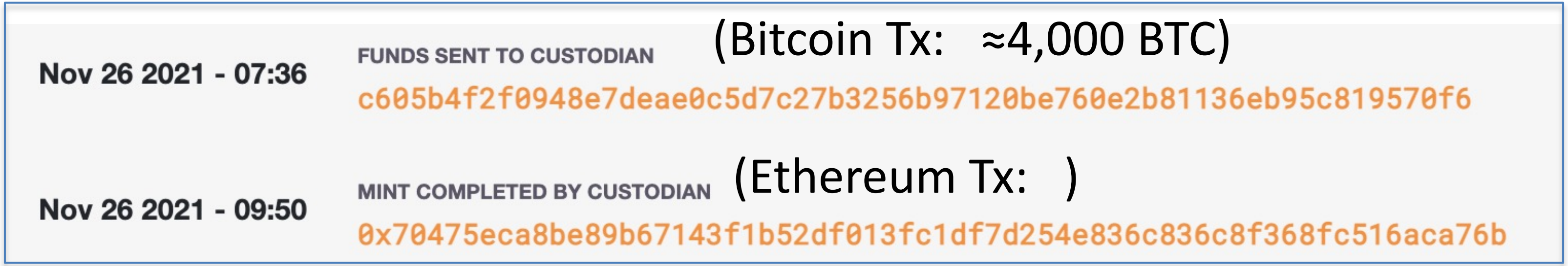
Alice wants her 1 BTC back

Moving 1 wBTC back to the Bitcoin network:



wBTC

Example BTC → Ethereum:



Nov 26 2021 - 07:36

FUNDS SENT TO CUSTODIAN (Bitcoin Tx: ≈4,000 BTC)

c605b4f2f0948e7deae0c5d7c27b3256b97120be760e2b81136eb95c819570f6

Nov 26 2021 - 09:50

MINT COMPLETED BY CUSTODIAN (Ethereum Tx:)

0x70475eca8be89b67143f1b52df013fc1df7d254e836c836c8f368fc516aca76b

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

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.

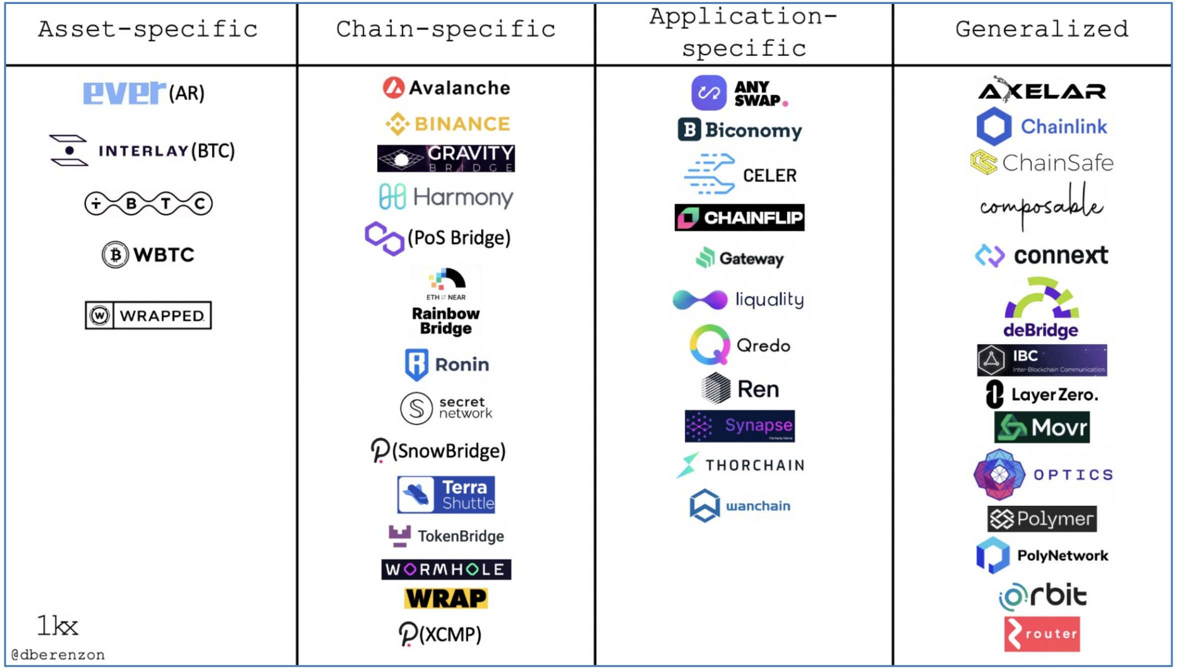
Custodians must lock 1.5x ETH stake for the BTC they manage

- 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



Asset-specific	Chain-specific	Application-specific	Generalized
ever (AR) INTERLAY (BTC) tBTC WBTC WRAPPED	Avalanche BINANCE GRAVITY BRIDGE Harmony (PoS Bridge) ETH NEAR Rainbow Bridge Ronin secret network (SnowBridge) Terra Shuttle TokenBridge WORMHOLE WRAP (XCMP)	ANY SWAP Biconomy CELER CHAINFLIP Gateway liquality Qredo Ren Synapse THORCHAIN wanchain	AXELAR Chainlink ChainSafe composable connext deBridge IBC Inter-Blockchain Communication LayerZero. Movr OPTICS Polymer PolyNetwork orbit router

1kx
@dberenzon

https://medium.com/1kxnetwork/blockchain-bridges-5db6afac44f8

Two types of bridges

Type 1: a lock-and-mint bridge

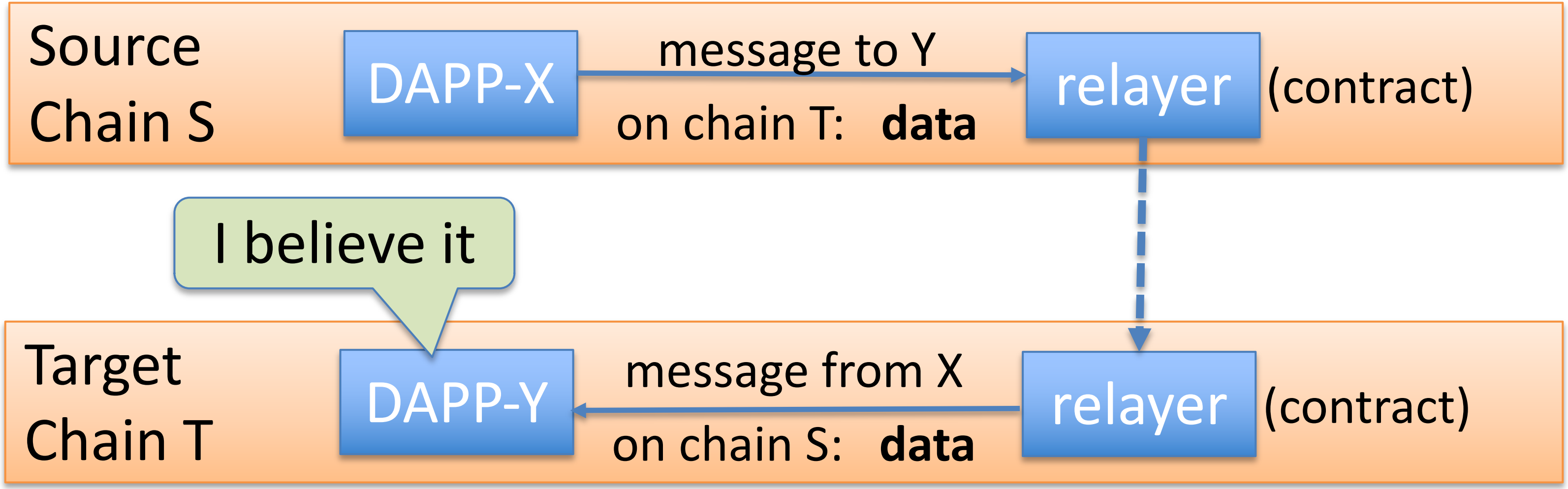
- SRC → DEST: user locks funds on SRC side,
  wrapped tokens are minted on the DEST side
- DEST → SRC: funds are burned on the DEST side,
  and released from lock on the SRC Side

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)

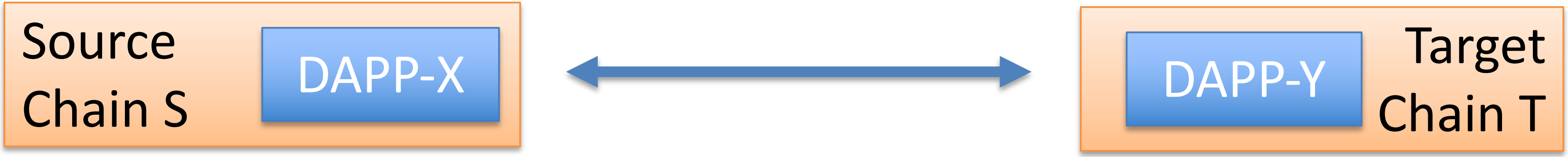
Step 1 (hard): a secure cross-chain messaging system



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

DAPP-Y

Target Chain T

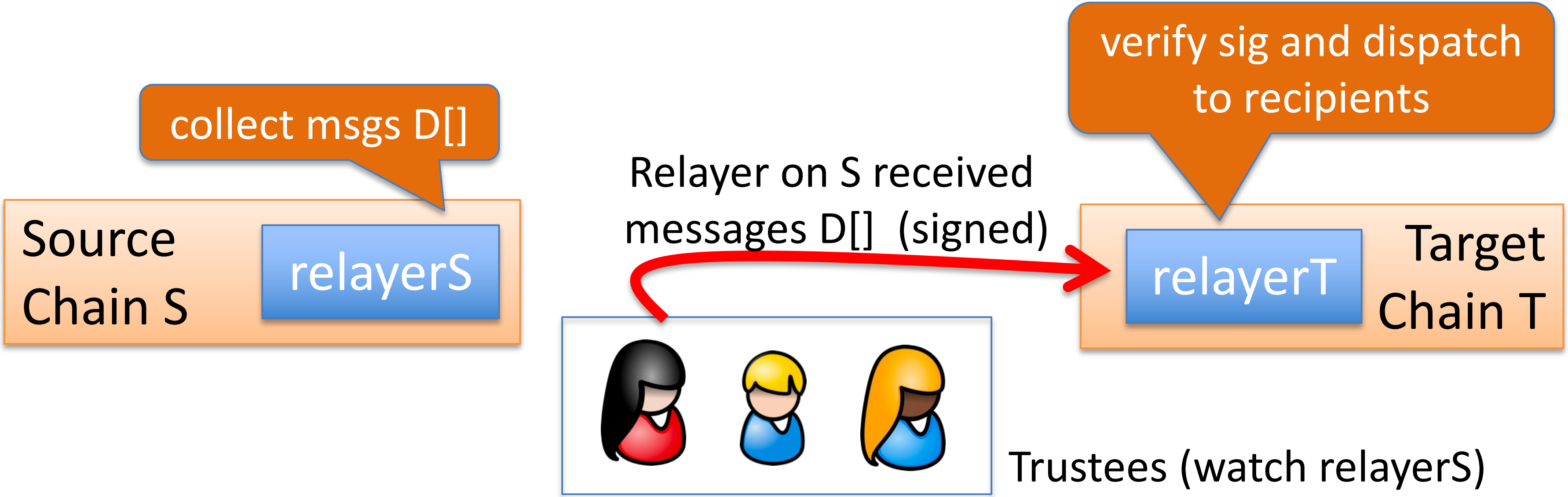
Step 2 (easier):   build a bridge using messaging system

- 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



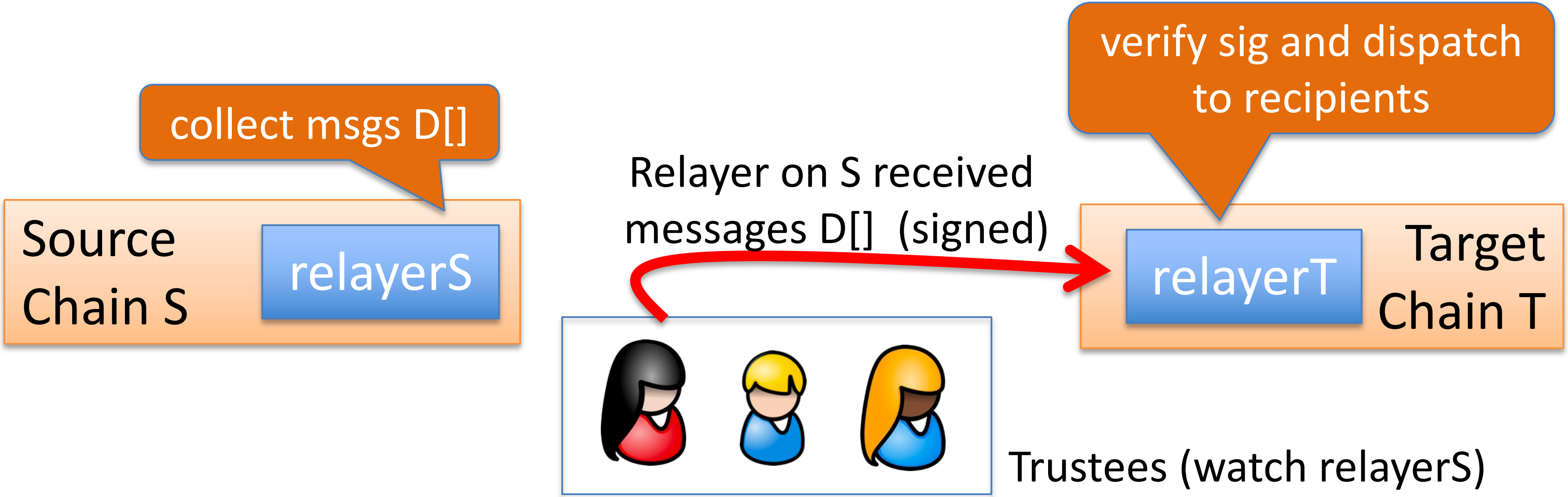
collect msgs D[]
Source Chain S
relayerS
verify sig and dispatch to recipients
Relayer on S received messages D[] (signed)
relayerT
Target Chain T
Trustees (watch relayerS)

RelayerT dispatches only if all trustees signed

⟹ **<u>if</u>** 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



collect msgs D[]

Source Chain S

relayerS

Relayer on S received messages D[] (signed)

verify sig and dispatch to recipients

relayerT

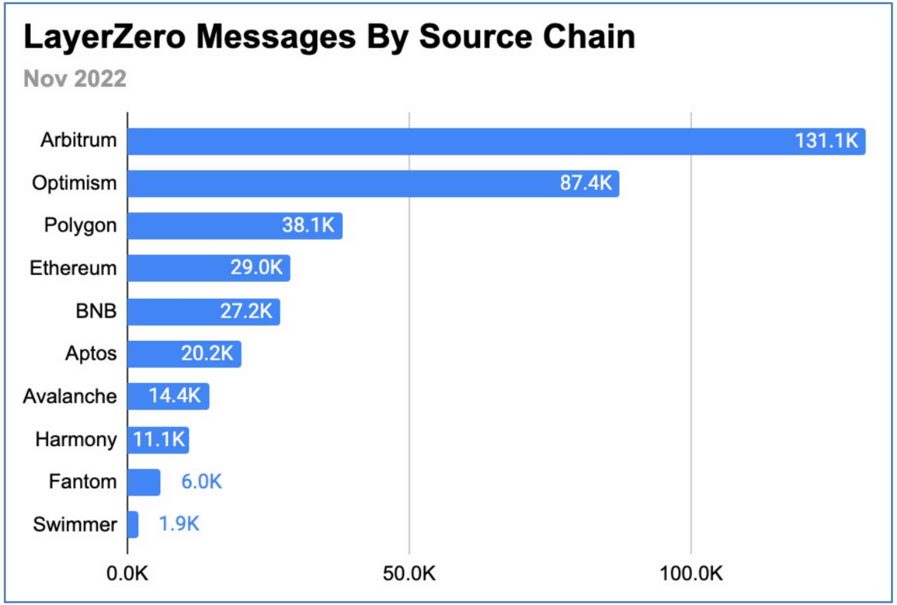
Target Chain T

Trustees (watch relayerS)

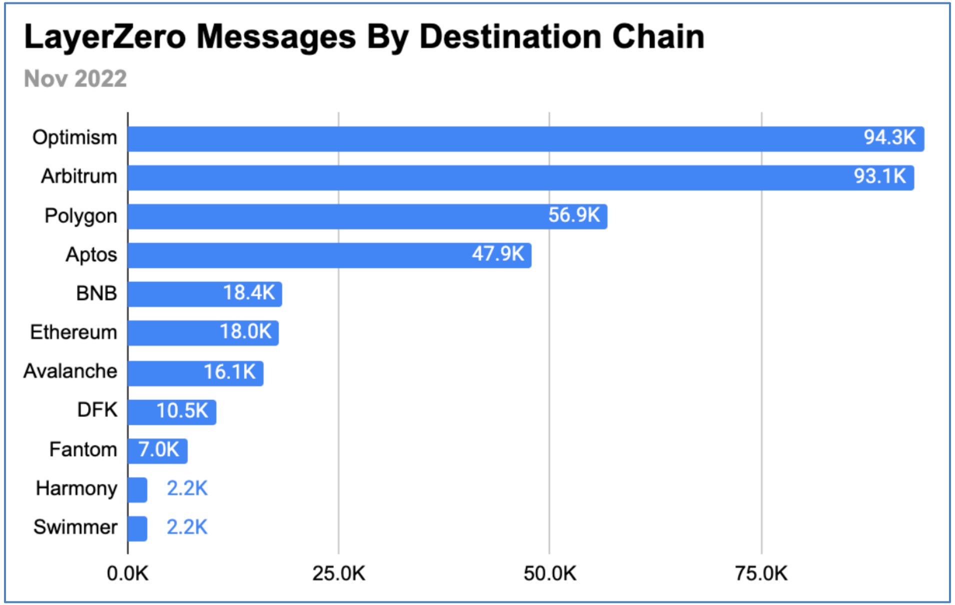
What if trustees sign and post a fake message to relayerT?

- off-chain party can send trustee’s signature to relayerS ⟹ trustee slashed

Activity



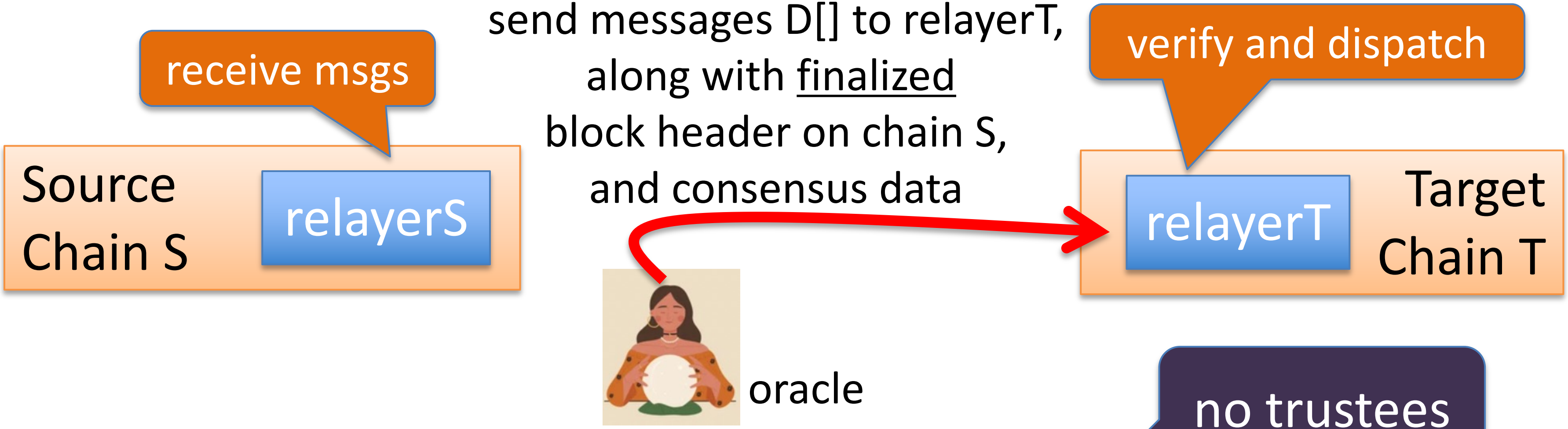
LayerZero Messages By Source Chain
Nov 2022
Arbitrum
131.1K
Optimism
87.4K
Polygon
38.1K
Ethereum
29.0K
BNB
27.2K
Aptos
20.2K
Avalanche
14.4K
Harmony
11.1K
Fantom
6.0K
Swimmer
1.9K
0.0K
50.0K
100.0K



LayerZero Messages By Destination Chain
Nov 2022
Optimism
94.3K
Arbitrum
93.1K
Polygon
56.9K
Aptos
47.9K
BNB
18.4K
Ethereum
18.0K
Avalanche
16.1K
DFK
10.5K
Fantom
7.0K
Harmony
2.2K
Swimmer
2.2K
0.0K
25.0K
50.0K
75.0K

Primarily two types of messaging systems

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

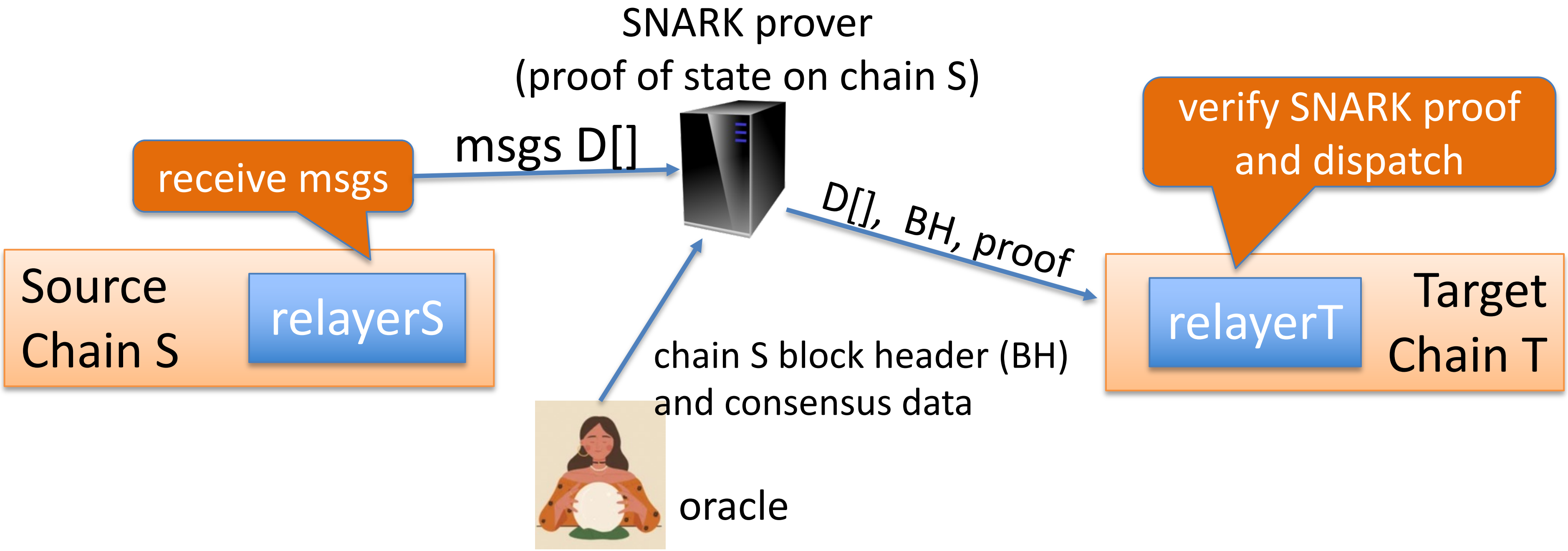

receive msgs
Source Chain S
relayerS
send messages D[] to relayerT, along with finalized block header on chain S, and consensus data
verify and dispatch
relayerT
Target Chain T
oracle

relayerT runs a (light) client for chain S to verify that relayerS received messages D[]

no trustees

assumes security of light client

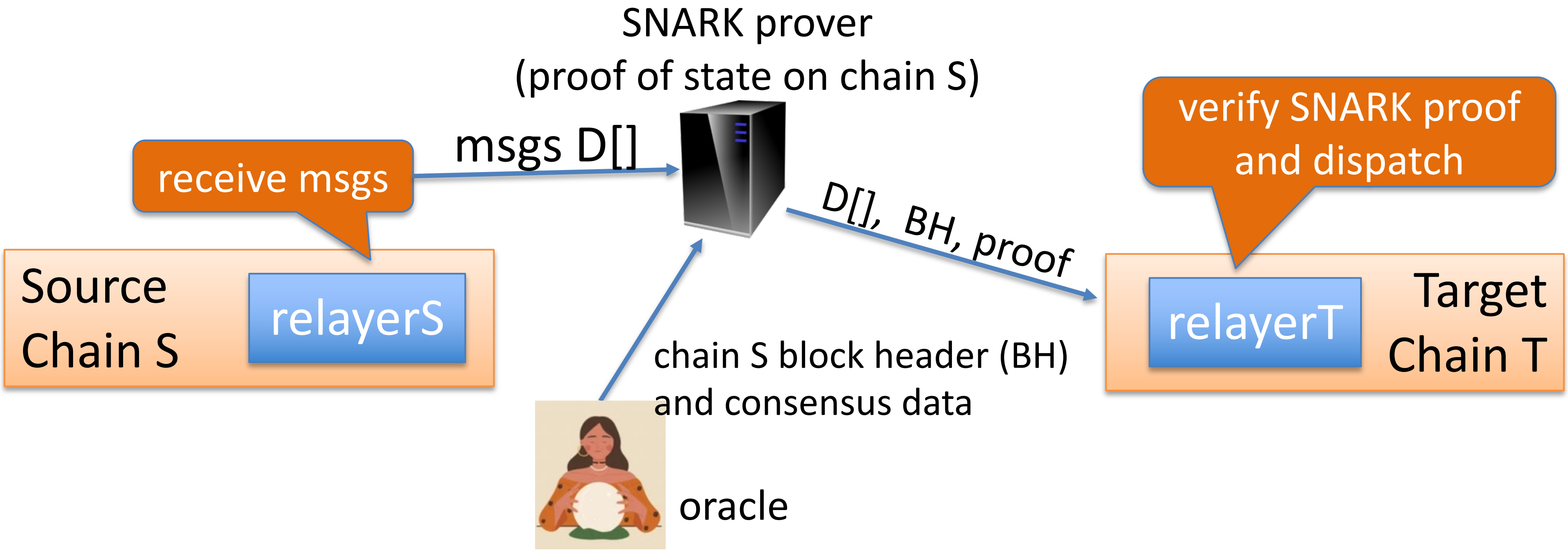
Primarily two types of messaging systems


SNARK prover
(proof of state on chain S)
verify SNARK proof and dispatch
msgs D[]
receive msgs
D[], BH, proof
Source Chain S
relayerS
relayerT
Target Chain T
chain S block header (BH) and consensus data
oracle

Problem: high gas costs on chain T to verify state of source chain S.

Solution: zkBridge: use SNARK to reduce work for relayerT

Primarily two types of messaging systems



SNARK prover
(proof of state on chain S)

receive msgs

msgs D[]

verify SNARK proof
and dispatch

D[], BH, proof

Source
Chain S

relayerS

relayerT

Target
Chain T

chain S block header (BH)
and consensus data

oracle

… being built by Succinct Labs

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

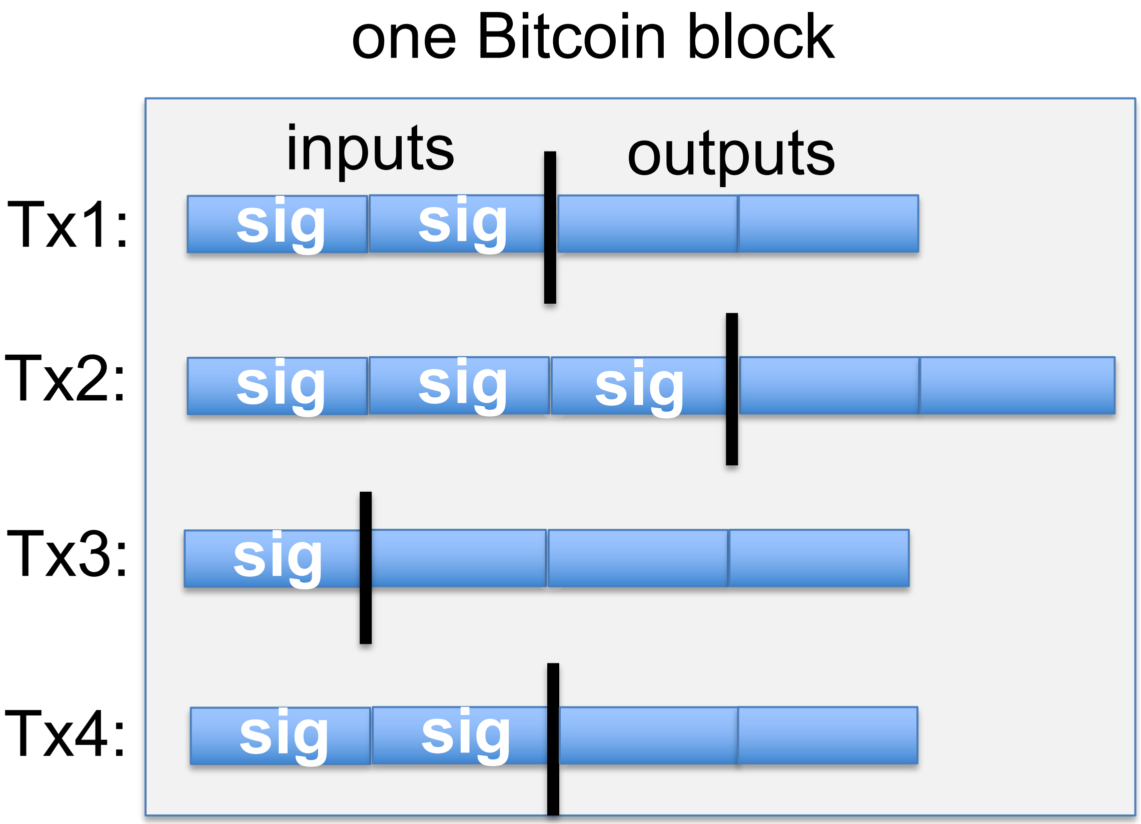
We are not there yet …

END OF LECTURE

Next lecture: super cool final guest lecture

Fun crypto tricks

BLS signatures



Signatures make up most of Tx data.

Can we compress signatures?

- Yes: aggregation!
- not possible for ECDSA

BLS Signatures

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

The setup:

- $G = \{1, g, \ldots, g^{q-1}\}$ a cyclic group of prime order q
- $H: M \times G \to G$ a hash function (e.g., based on SHA256)

BLS Signatures

KeyGen(): choose random α in $\{1, \ldots, q\}$

output $\boxed{\text{sk} = \alpha \;, \quad \text{pk} = g^{\alpha} \quad \in \text{G}}$

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

Verify(pk, m, sig): output accept if $\log_{g}(\text{pk}) = \log_{H(m,\text{pk})}(\text{sig})$

Note: signature on m is unique! (no malleability)

How does verify work?

A pairing: an efficiently computable function e:G × G → G'

such that $\boxed{e(g^{\alpha}, g^{\beta}) = e(g,g)^{\alpha\beta}}$ for all $\alpha, \beta \in \{1, \ldots q\}$

and is not degenerate: $e(g,g) \neq 1$

verify test

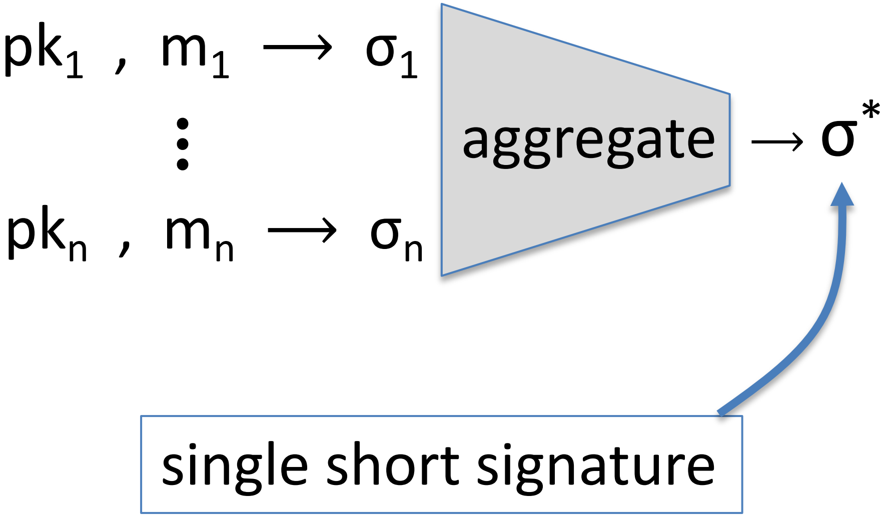
Observe: $\log_{g}(pk) = \log_{H(m,pk)}(sig)$

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

$$\begin{array}{ccc} \shortparallel & \qquad & \shortparallel \\ e(g, H(m,pk)^{\alpha}) & = & e(g^{\alpha}, H(m,pk)) \end{array}$$

Properties: signature aggregation [BGLS'03]

Anyone can compress n signatures into one



pk_1 , m_1 $\longrightarrow$ σ_1

⋮

pk_n , m_n $\longrightarrow$ σ_n

aggregate $\rightarrow \sigma^*$

single short signature

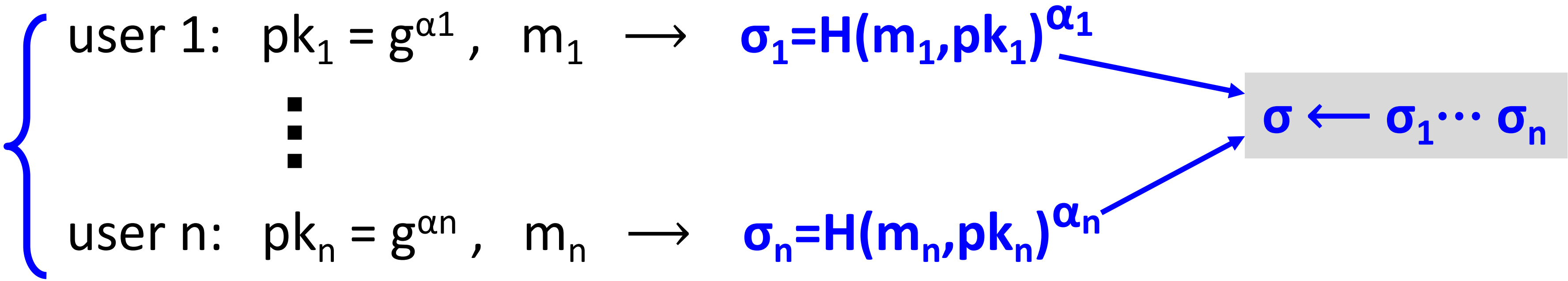
Verify($\overline{\mathbf{pk}}$, $\overline{\mathbf{m}}$, σ^*) = "accept"

convinces verifier that

for i=1,…,n:

user i signed msg m_i

Aggregation: how



user 1: $pk_1 = g^{\alpha 1}$, m_1 ⟶ $\sigma_1 = H(m_1, pk_1)^{\alpha_1}$

⋮

user n: $pk_n = g^{\alpha n}$, m_n ⟶ $\sigma_n = H(m_n, pk_n)^{\alpha_n}$

$\sigma \longleftarrow \sigma_1 \cdots \sigma_n$

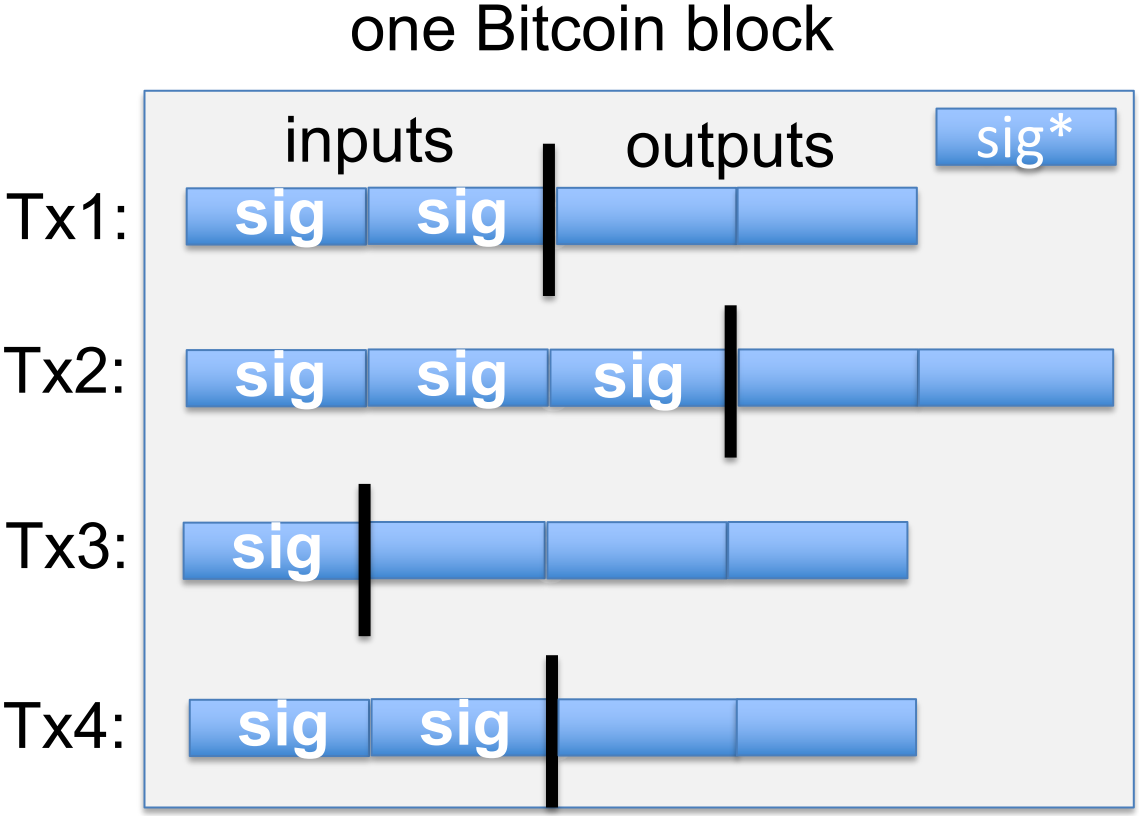
Verifying an aggregate signature: (incomplete)

$$\prod_{i=1}^{n} e(H(mi,pki), g^{\alpha_i}) \overset{?}{=} e(\boldsymbol{\sigma}, g)$$

$$/\!/ \qquad\qquad\qquad\qquad \backslash\!\backslash$$

$$\Pi_{i=1}\, e(H(m_i,pk_i)^{\alpha_i}, g) \;=\; e(\Pi_{i=1} H(m_i,pk_i)^{\alpha_i}, g)$$

Compressing the blockchain with BLS



<ins>if needed</ins>:

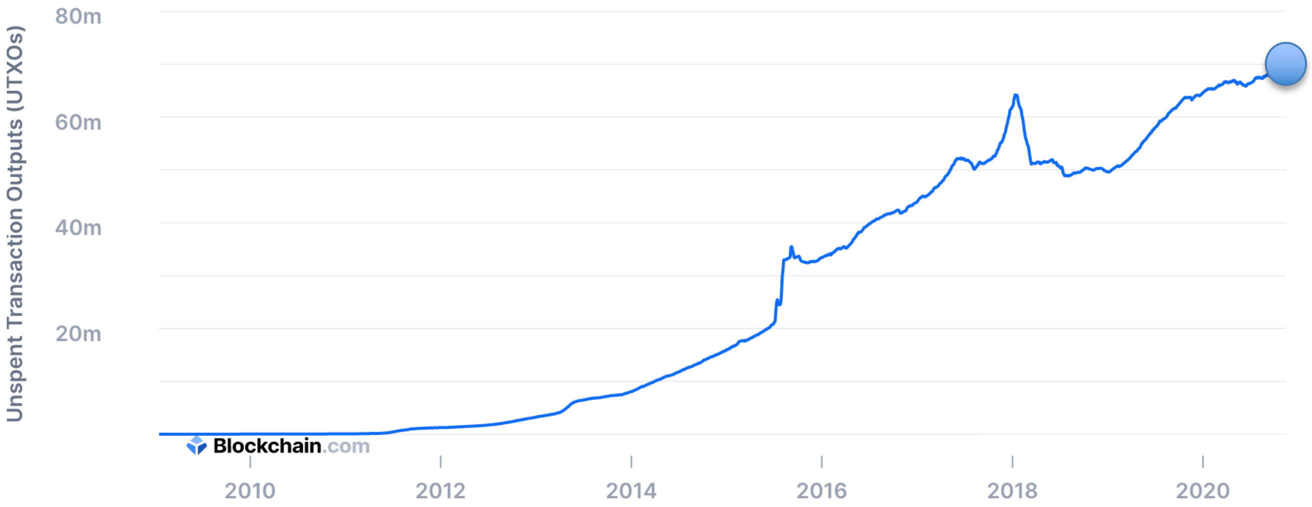
compress all signatures in a block into a single aggregate signatures

⇒ shrink block

or: aggregate in smaller batches

Reducing Miner State

UTXO set size



≈70M UTXOs

Miners need to keep all UTXOs in memory to validate Txs

Can we do better?

Recall: polynomial commitments

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

- *eval*: goal: for a given ***com$_f$*** and x, y ∈ $\mathbb{F}_p$,

  construct a SNARK to prove that f(x) = y.

Homomorphic polynomial commitment

A polynomial commitment is **homomorphic** if

there are efficient algorithms such that:

- *commit*(*pp*, f_1, r_1) → $\boldsymbol{com_{f1}}$ *commit*(*pp*, f_2, r_2) → $\boldsymbol{com_{f2}}$

Then:

(i) for all $a, b \in \mathbb{F}_p$: $\boldsymbol{com_{f1}}$, $\boldsymbol{com_{f2}}$ → $\boldsymbol{com_{a*f1+b*f2}}$

(ii) $\boldsymbol{com_{f1}}$ → $\boldsymbol{com_{X*f1}}$

Committing to a set (of UTXOs)

(accumulator)

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

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

Set: $\boldsymbol{com}_f$ = 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: $\boldsymbol{com}_f \rightarrow \boldsymbol{com}_{X*f-U*f}$ ⇽ short commitment to $S \cup \{U\}$

How does this help?

Miners maintain two commitments:

$$\left.\begin{array}{ll}\text{(i)} & \text{commitment to set T of all UTXOs}\\ \text{(ii)} & \text{commitment to set S of spent TXOs}\end{array}\right\} \leq 1\text{KB}$$

com$_T$, com$_S$

Tx format:

- every input U includes a proof $(U \in T \ \ \&\& \ \ \mathrm{U} \notin S)$

  Two eval proofs: $T(U) = 0 \ \ \&\& \ \ S(U) \neq 0$ (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 ??

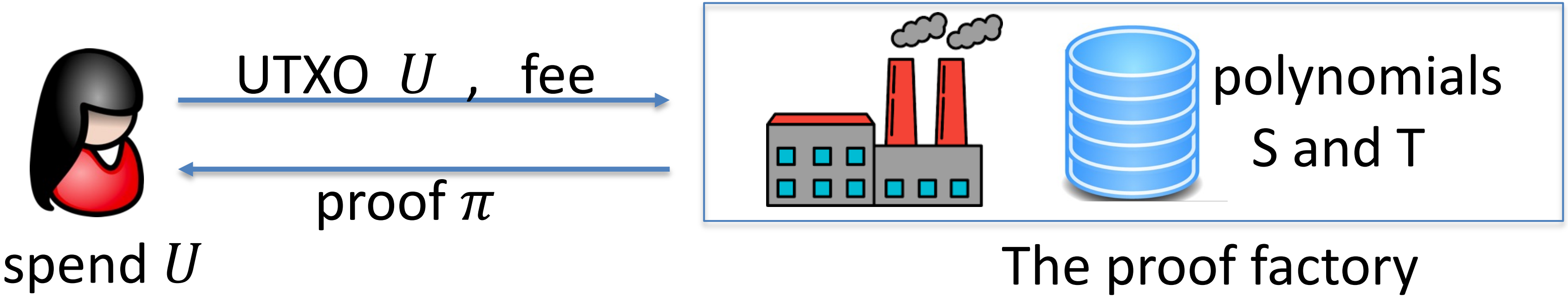
Problem: how does a user prove that her UTXO U satisfies

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

This requires knowledge of the entire blockchain

⇒ user needs large memory and compute time

⇒ … can be outsourced to an untrusted 3rd party



UTXO U , fee

proof π

spend U

polynomials S and T

The proof factory

Is this practical?

Not quite …

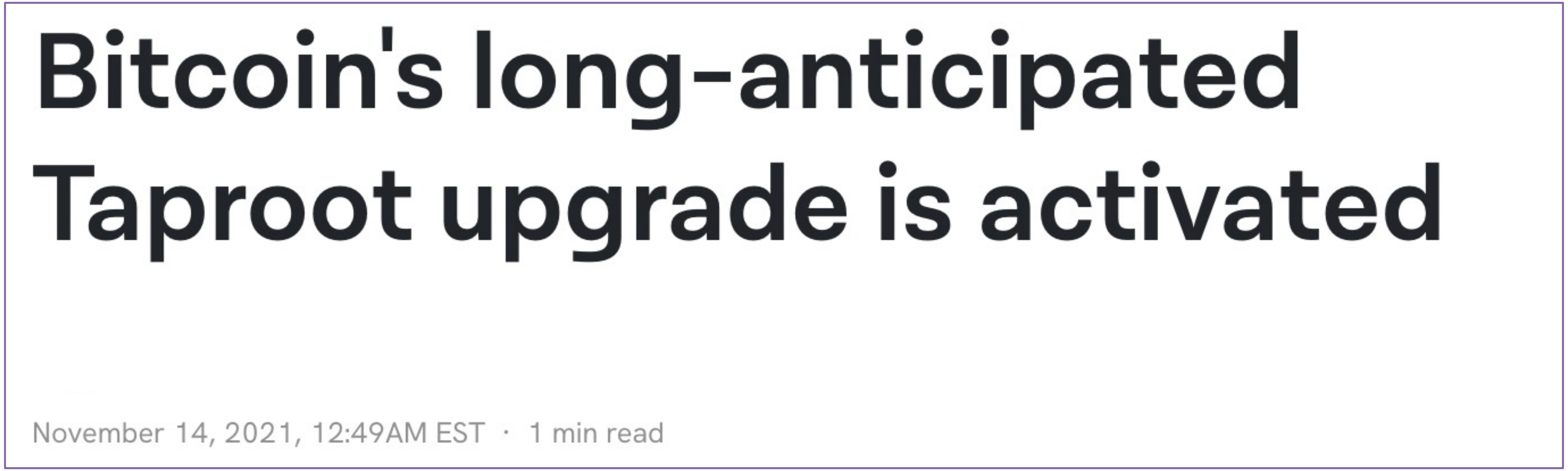
- Problem: the factory's work per proof is linear in the number of UTXOs ever created

- Many variations on this design:
  - can reduce factory's work to $\log_2$(# 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 …

How?

ECDSA and Schnorr public keys:

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

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

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

  ⇒ an interactive protocol between Alice and Bob

  (note: much simpler with BLS)

  ⇒ 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,

  $\Rightarrow$ 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