CS251 Fall 2020

(cs251.stanford.edu)

Recursive SNARKs

Benedikt Bünz

Hitchhikers guide to the galaxy

What if we want to verify that computation?

Input → Output (42)

Long Computation

Recap: Non-interactive Proof Systems

A **non-interactive proof system** is a triple (S, P, V):

- **S**(C) $\rightarrow$ public parameters (S_p, S_v) for prover and verifier

  (S_p, S_v) is called a *reference string*

- **P**(S_p, $\boldsymbol{x}$, $\boldsymbol{w}$) $\rightarrow$ proof π

- **V**(S_v, $\boldsymbol{x}$, $\boldsymbol{\pi}$) $\rightarrow$ accept or reject

SNARKs for long computations

Issues:

-P takes very long

-Starts after proving ***after*** computation finished

-Can't hand off computation

-S also runs at least linear in |C|

(ok if many proofs)

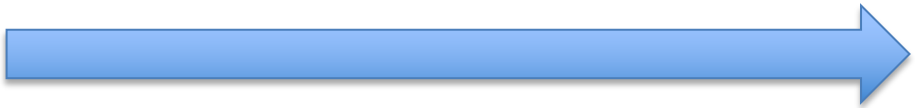
C – Circuit for long computation

$\mathbf{S}(C) \rightarrow (S_p, S_v)$

$x = (input, output)$

$w = transcript$

Input



Long Computation, Transcript

Output (42)

$\mathbf{P}(S_p, x, w) \rightarrow \pi$

$\mathbf{V}(S_v, x, \pi) \rightarrow$ accept

Handing off computation

C_I– Circuit for ~~long~~ intermediate computation

$\mathbf{S}(C_I) \rightarrow (S_p, S_v)$

$x_1 = (input, int_1), w_1 = transcript_1$

$x_2 = (int_1, int_2), w_2 = transcript_2$

$x_3 = (int_2, output), w_3 = transcript_3$

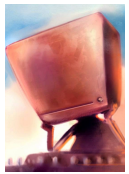
$\mathbf{P}(S_p, x_i, w_i) \rightarrow \pi_i$

$\mathbf{V}(S_v, x_1, \pi_1)$

$\mathbf{V}(S_v, x_2, \pi_2)$

$\mathbf{V}(S_v, x_3, \pi_3)$

$|\pi|$/ **V** linear in #handoffs



Input → Int_1, π_1 → Int_2, π_2 → Output (42), π_3

$transcript_1$ $transcript_2$ $transcript_3$

Incremental Proofs

- We need updatable/incremental proofs

C_I– Circuit per computation step, t number of steps/handoffs

S(C_I) → (S_p, S_v)

P(S_p, $\boldsymbol{x_i}$, $\boldsymbol{w_i}$, π_{i-1}) → updated proof π_i // $\pi_0 = \perp$

V(S_v, $\boldsymbol{x_0}$, $\boldsymbol{x_t}$, π_t, t) → accept/reject

$|\pi_i| = |\pi_{i-1}|$ // proofs don't grow

PhotoProof

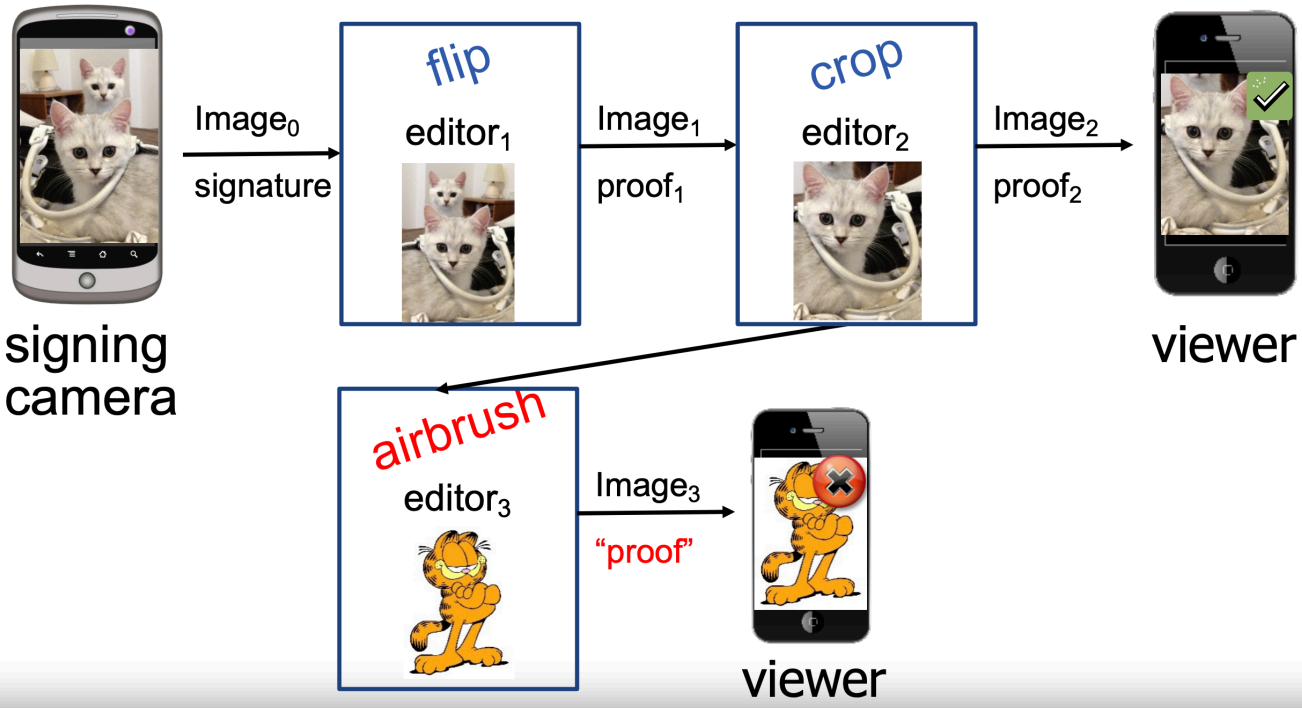
signing camera

Viewer can still verify authenticity

viewer

Allow valid updates of photo and provide proof

PhotoProof



Proof allows valid edits only, Incrementally updated

Recursive Proofs

- How can we build incremental proofs?

"Proof of a proof":
A proof that π_2 that I know a proof π_1 that $C(x_1, w_1) = 0$

"Proof of a proof of a proof …":
A proof that π_2 that I know a proof π_1 which proves knowledge of a proof π_0 that $C(x_0, w_0) = 0$

SNARK of SNARK

$$S(C) \rightarrow S_P, S_V$$
$$\pi \leftarrow P(x, w)$$

Now write a circuit C' that verifies π:

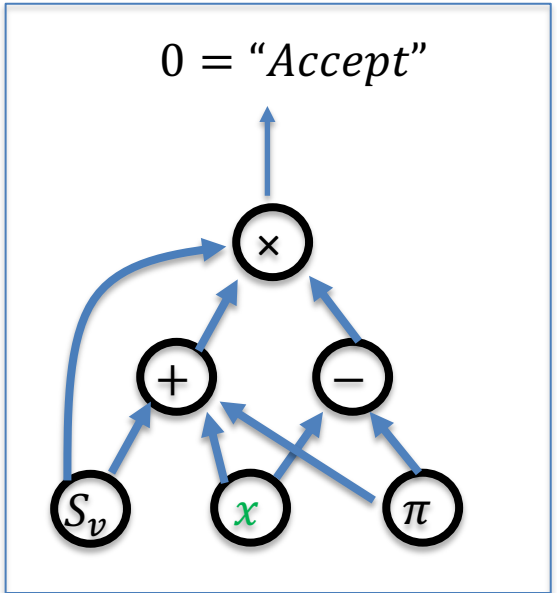
- Input x' is x
- Witness w' is π
- $C'(x'$, w') = 0 iff V(S_V, π, x)=Accept

Finally:

$$S(C') \rightarrow S'_P, S'_V$$
$$\pi' \leftarrow P(x', w')$$

SNARK of SNARK…

- Note that C’ depends only on **V** and S_v
- We can express **V** as a circuit:



SNARK of SNARK…

- We can also make C' more complex…
  - Input x' is x_0, x_1
  - Witness $w'$is π, w_1
  - $C'(x', w') = 0$ iff $V(S_V, \pi, x_0) = Accept$ AND $C(x_1, w_1) = 0$

SNARK of SNARK…

- We can also make C' more complex…
  - Input x' is ~~x_0~~, x_1
  - Witness w' is π, w_1, x_0
  - $C'(x', w') = 0$ iff $V(S_V, \pi, x_0) = Accept$ AND $C(x_0, x_1, w_1) = 0$

C implements $x_1 = F(x_0)$

- What about proving verification of π'?
- Needs new C'' that has S'_V hardcoded?
- Do we need to re-run setup for every additional level of recursion?
- In some applications we can use the same S_V over and over again
  - Repeated application of single function

SNARK of SNARK…

- Do verifiers need to re-run setup for every additional level of recursion? **Proposed solution: make S_V part of the witness**
- Modify definition C'_i:
  - Input x' is $x_0, x_1, \ldots, x_i$
  - Witness w' is π, w_i, S_V
  - $C'(x', w') = 0$ iff $V(S_V, \pi, x_0, \ldots, x_{i-1}) = Accept$ AND $S(C'_{i-1}) \rightarrow (S_P, S_V)$ AND $C(x_i, w_i) = 0$
- Now C'_i may accept as witness a proof π that is a proof for C'_{i-1} generated using parameters $\left(S_P^{i-1}, S_V^{i-1}\right) \leftarrow S(C'_{i-1})$. *Prover still needs to re-run setup as part of proving.*

Universal SNARK Verifier

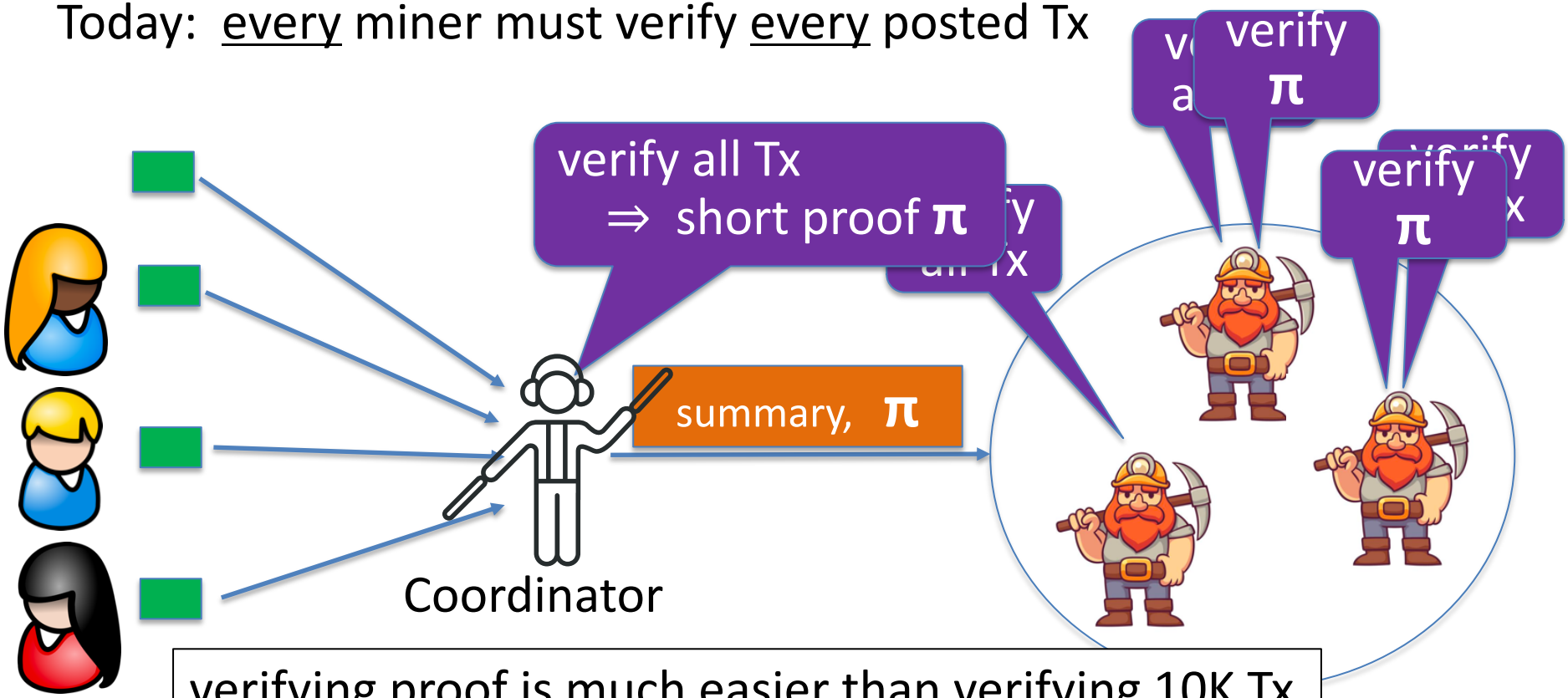
- What goes wrong when the setup is trusted? **Making S_V a witness does not work.**
  - S_V "exists" even for proofs of false statements
  - Proof system only sound when prover does not know the secrets involved in the generation of S_V.
  - The existence of (S_V, π) that the algorithm V would accept is meaningless.
- New solution: **universal SNARK verifier**

Universal SNARK verifier

- UC is a circuit that takes inputs (C, x, w) and outputs $C(x, w)$
- $S(UC) \rightarrow (US_P, US_V)$ are parameters of SNARK system for UC
- Think of UC as an x86 processor
  - Takes in input and instructions and executes them
- Define C_i':
    - Input x' is $x_0, x_1, \ldots, x_i$
    - Witness w' is π, w_i, S_V
    - $C'(x', w') = 0$ iff $V(US_V, \pi, C_{i-1}', x_0, \ldots, x_{i-1}) = Accept$ AND AND $C(x_i, w_i) = 0$

Recap: SNARKRollup

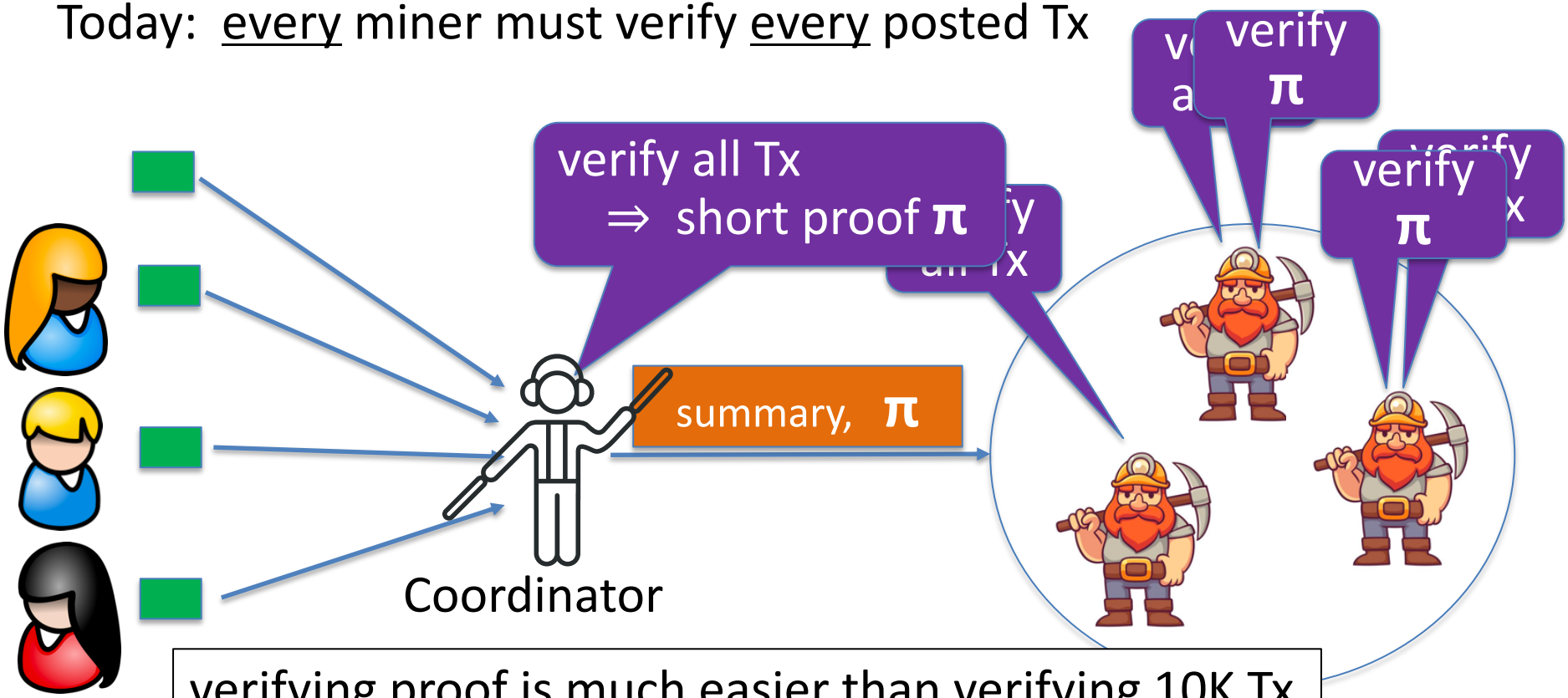
Today: <u>every</u> miner must verify <u>every</u> posted Tx



verifying proof is much easier than verifying 10K Tx

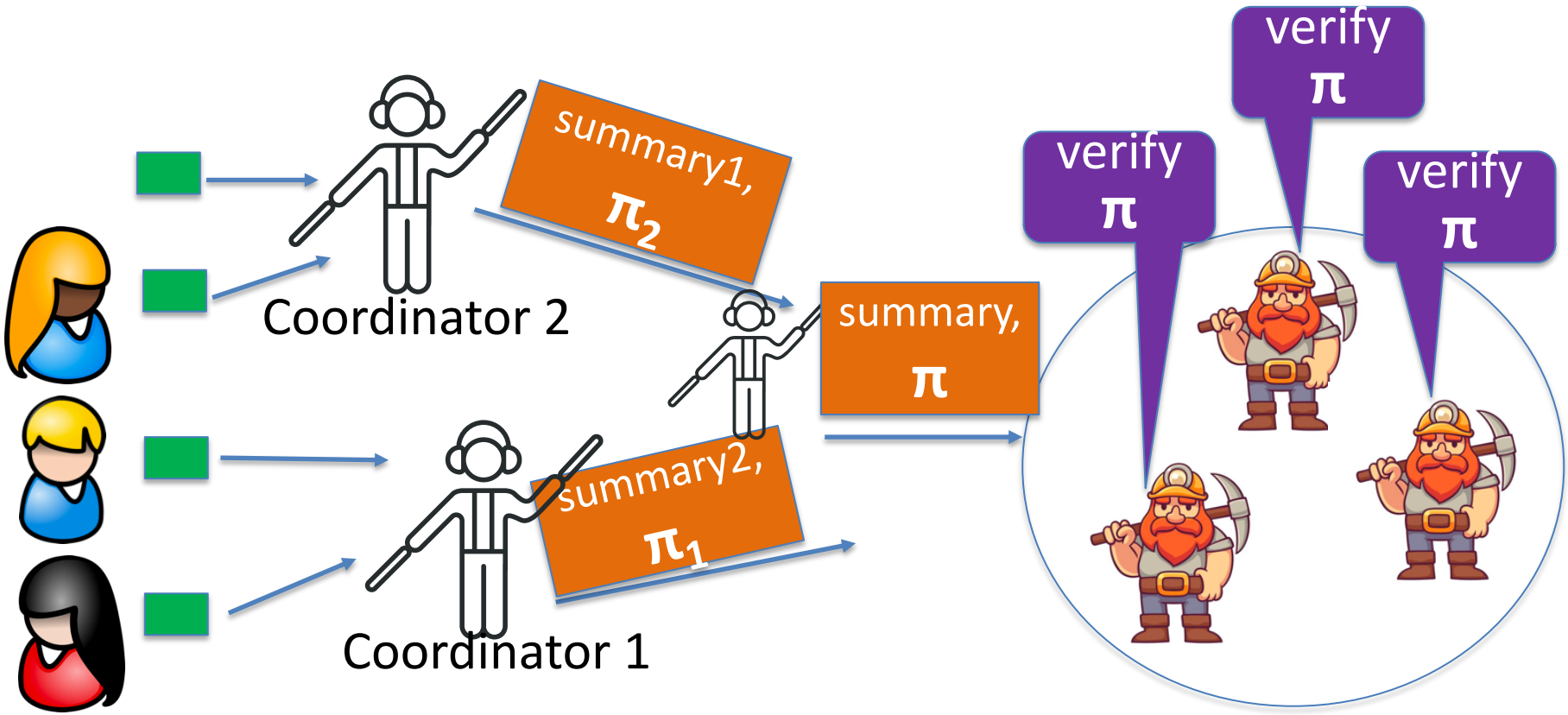
Recap: Rollup

Today: <u>every</u> miner must verify <u>every</u> posted Tx



verifying proof is much easier than verifying 10K Tx

Rollup with many coordinators

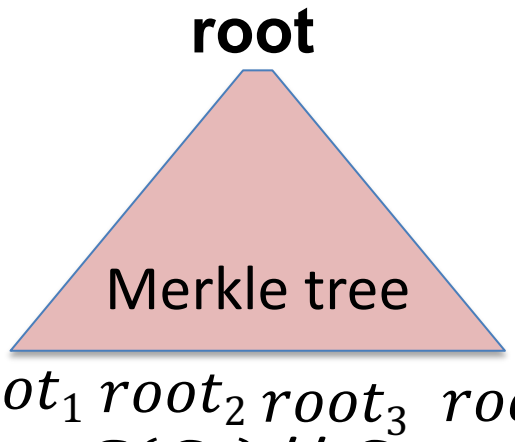


SNARK^2-Rollup

- Multiple coordinators/servers
- Each responsible for subset of users (no overlaps)
- Super coordinator (can be one of the coordinators)
- Super coordinator combines summaries (balance table) and creates one proof that

SNARK^2-Rollup

- Let **root$_i$** be the Merkle Tree Root of summary i

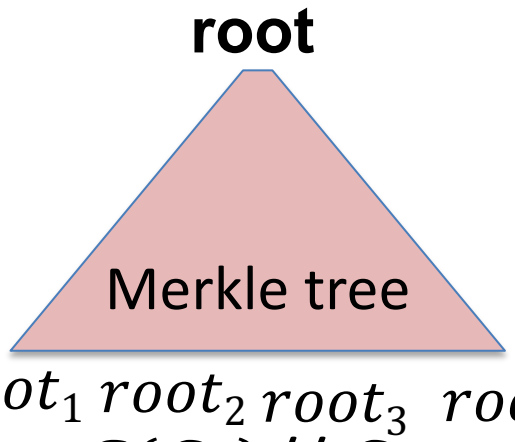


Build one root from summaries

- $S_V, S_P \leftarrow \boldsymbol{S}(C_C)$// C_C coordinator circuit
- $C_{SC}(\mathrm{x} = S_V, \mathbf{root};\ \mathrm{w} = root_1, root_2 \ldots, \pi_1, \pi_2, \ldots)$:
- **V**$(S_V, x = root_i, \pi_i)$ for all i and **root**=MT($root_i$s)

SNARK^2-Rollup

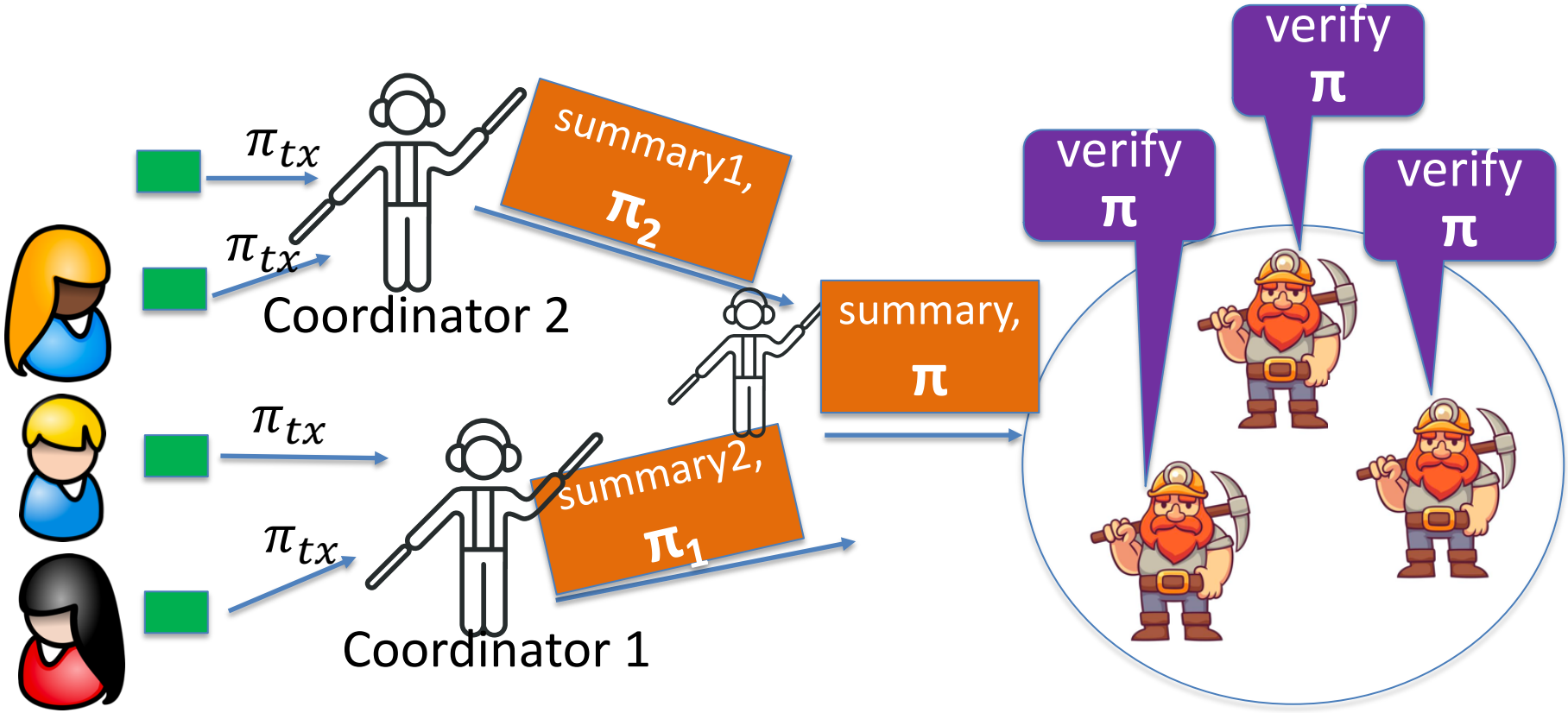
- Let **root$_i$** be the Merkle Tree Root of summary i



Build one root from summaries

- $S_V, S_P \leftarrow \boldsymbol{S}(C_C)$// C_C coordinator circuit
- $C_{SC}(\mathrm{x} = S_V, \mathbf{root};\ \mathrm{w} = root_1, root_2 \ldots, \pi_1, \pi_2, \ldots)$:
- **V**$(S_V, x = root_i, \pi_i)$ for all i and **root**=MT($root_i$s)

SNARK³-Rollup



Enhancing transactions with SNARKs

- We've seen that private transactions require zero-knowledge proofs
- Add ZK-SNARKs to every transaction
- Level 1 coordinators verify transaction by verifying transaction ZK-SNARKs
- Additionally we can have more complicated transactions (Smart Contracts)
  - Transaction verification is constant time regardless of proof complexity

SNARK^3-Rollup

- Smart Contract SC_i
- C_t = Commit(Smart Contract State t,r_t)
- C_{t+1} = Commit(Smart Contract State t+1, r_{t+1})
- $S_{V_i}, S_{P_i} \leftarrow \boldsymbol{S}(ZK - SC_i)$ // Zero-Knowledge SC
- Key-root=MT(S_{V_i}s) // Merke tree of all verification keys
- $ZK - SC_i(x_i = C_t, C_{t+1}; w_i = states\ t, t+1, r_t, r_{t+1}$):
  - C_t, C_{t+1} commit to $states\ t, t+1$ and transition is valid

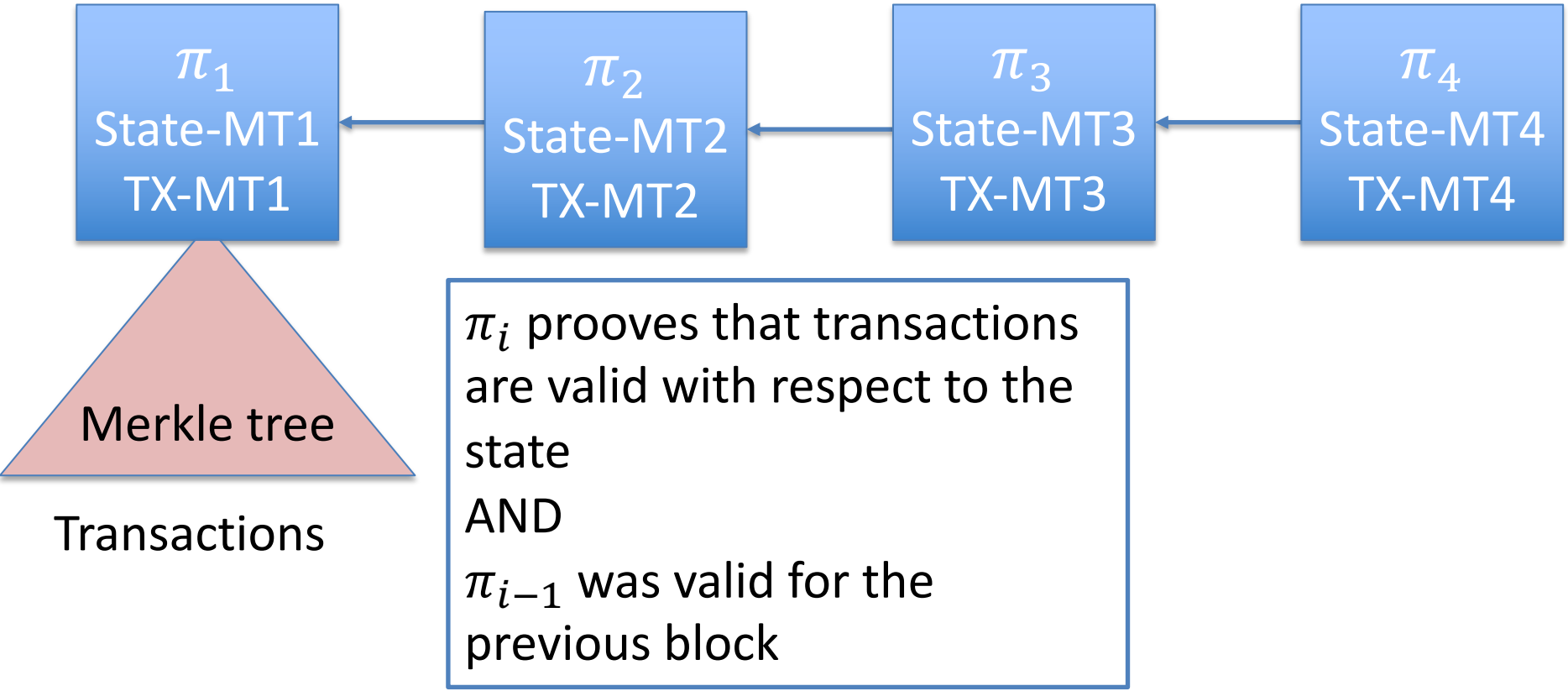
SNARK^3-Rollup

- Each user creates $\pi_i \leftarrow \boldsymbol{P}(S_{P_i}, x_i, \mathrm{w_i})$ **(ZK)** and outputs
  - $C_{t+1,i}, \pi_i$
- Level 1 coordinator takes many outputs and produces one proof using Key-root to select the correct verification key
- The coordinator does not care about which key is used just that it's the correct one
- Possible to hide state transition AND smart contract details

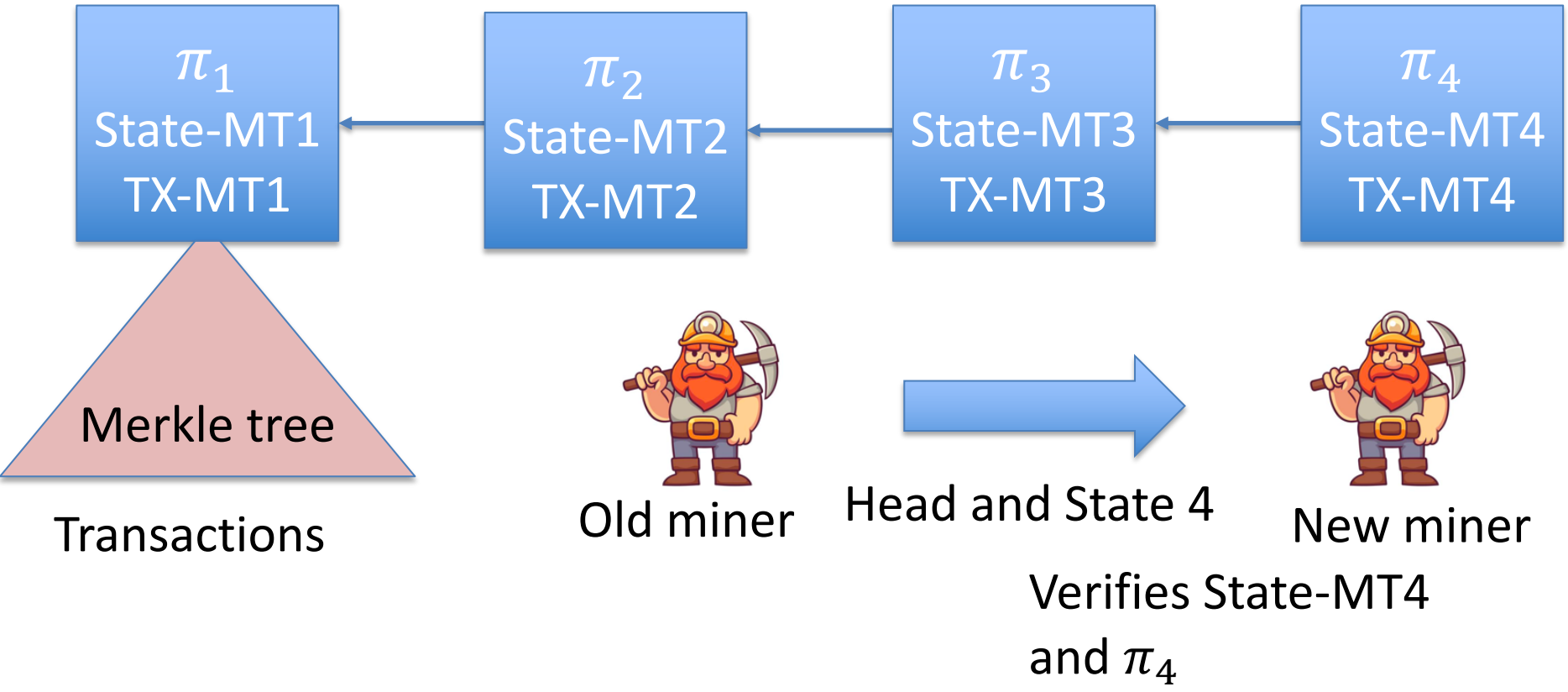
Constant size blockchains

- Rollup reduces the verification cost
- Still linear in the number of state updates
- When a node joins the network they need to verify one rollup proof per block!
- In general starting a full node requires verification of all blocks
  - Can take days!

Constant size Blockchain



Constant size Blockchain



Constant size Blockchain

- Light clients can verify every block!
  - Low memory, low computation
  - Independent of length of chain or #transactions
- Relies on data serving nodes for synching

- Practical today!

END OF LECTURE

Next lecture: Crypto tricks and open discussion

Please attend last two lectures if you can