Lecture 10: Polynomial Commitments with Short Proofs

Zero-knowledge proofs

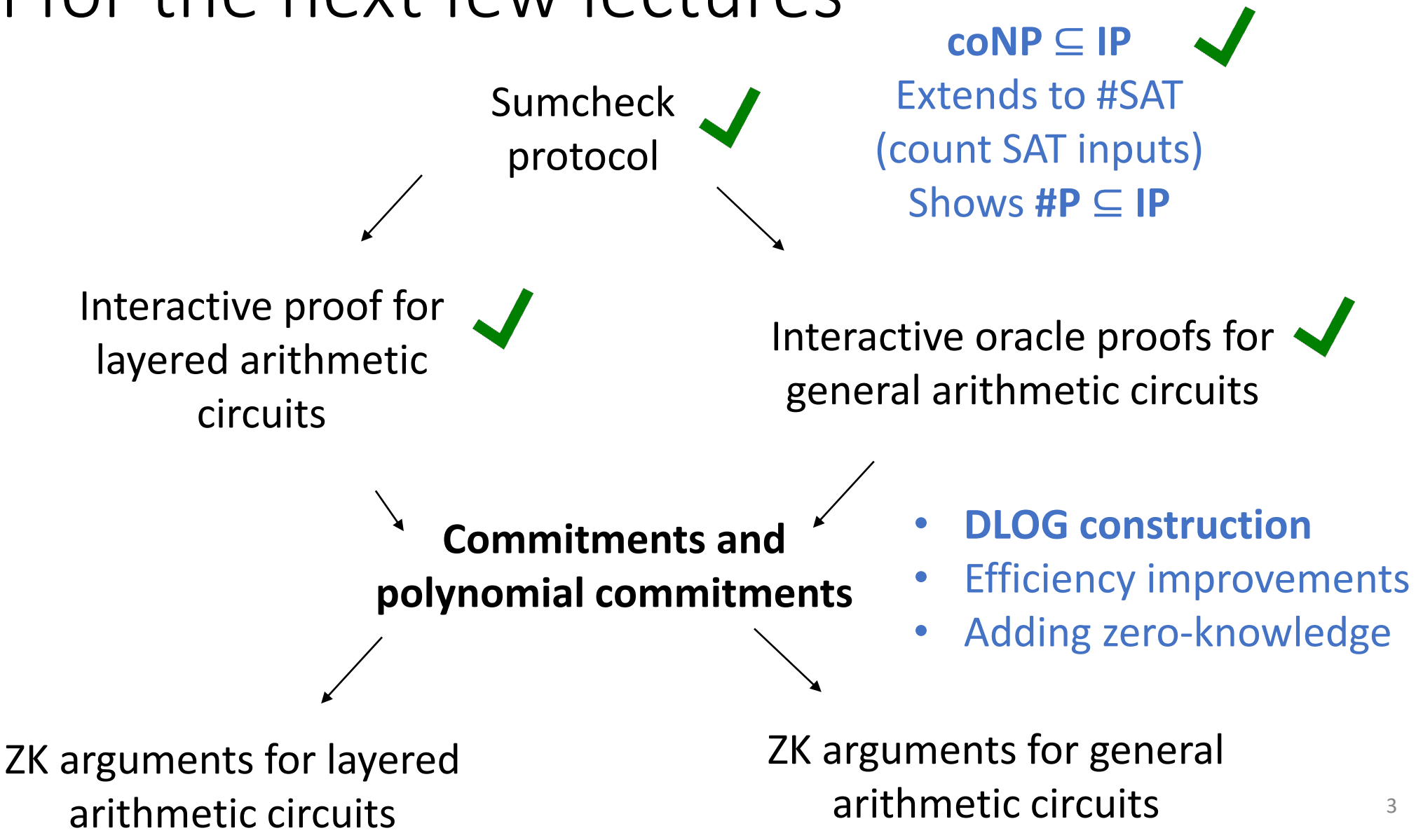
263-4665-00L

Lecturer: Jonathan Bootle

Announcements

- Graded homework due today, 24/11/2023 23:59 CET.
- Next Friday’s lecture 01/12/2023 in person again.

Plan for the next few lectures



Pedersen multi-commitments

Setup$(\mathbf{1}^{\lambda}, \boldsymbol{deg})$: Group $\mathbb{G}$ of prime order $p \approx 2^{\lambda}$, $G_0, \ldots, G_{N-1}, H \leftarrow_{\$} \mathbb{G}$. Output $pp := (\mathbb{G}, \vec{G} = (G_0, \ldots, G_{N-1}), H, p)$.

Commit$(\boldsymbol{pp}, \tilde{\boldsymbol{f}} \in \mathbb{Z}_{\boldsymbol{p}}[\boldsymbol{X}], \boldsymbol{r} \leftarrow_{\$} \mathbb{Z}_{\boldsymbol{p}})$:

Parse $\tilde{f}\left(\vec{X}\right) := \sum_{\vec{\imath} \in \{0,1\}^{log\ N}} f_{\vec{\imath}} \cdot \widetilde{\mathrm{EQ}}\left(\vec{X}; \vec{\imath}\right)$. Sample $r \leftarrow_{\$} \mathbb{Z}_p$.

$O(1)$ commitment size

Compute $(C, d) := \left(\sum_{\vec{\imath} \in \{0,1\}^{log\ N}} f_{\vec{\imath}} \cdot G_i + r \cdot H, r\right) = \left(\langle \vec{f}, \vec{G} \rangle + r \cdot H, r\right)$. Output (C, d).

Verify$(\boldsymbol{pp}, \boldsymbol{C}, \boldsymbol{d}, \boldsymbol{m})$: Output $C == \sum_{\vec{\imath} \in \{0,1\}^{log\ N}} f_{\vec{\imath}} \cdot G_i + r \cdot H$.

Eval: Given $y_1, \ldots, y_{\log N}, z \in \mathbb{F}$, c, prove knowledge of openings $\{f_{\vec{\imath}}\}, r$ of C such that $\sum_{\vec{\imath} \in \{0,1\}^{log\ N}} f_{\vec{\imath}} \cdot \widetilde{\mathrm{EQ}}(\vec{y}; \vec{\imath}) = z$. Use Σ-protocol techniques $O(N)$ proof size, V complexity

$$\mathcal{R}_{PedPC}(pp) := \left\{ \left((C, z, y_1, \ldots, y_\ell), \left(\tilde{f}, r\right) \right) : \begin{array}{c} \tilde{f} \in \mathbb{F}^{\leq 1}[X_1, \ldots, X_\ell] \\ \tilde{f}(y_1, \ldots, y_\ell) = z, C = \langle \vec{f}, \vec{G} \rangle + r \cdot H \end{array} \right\}$$

Eval protocol for Pedersen multicommitments

- Let $y_1, \dots, y_{\log N} \in \mathbb{Z}_p$ with $\tilde{f}(y_1, \dots, y_{\log N}) = z$.
- Let $\vec{Y} := \mathrm{Expand}(\vec{y}) = \left(\widetilde{\mathrm{EQ}}(\vec{y}; \vec{\imath})\right)_{\vec{\imath} \in \{0,1\}^{\log N}}$ satisfying $f(\vec{y}) = \langle \vec{f}, \vec{Y} \rangle$.

Example:

$$\tilde{f}(X_1, X_2) = f_{00}(1 - X_1)(1 - X_2) + f_{01}X_1(1 - X_2) + f_{10}(1 - X_1)X_2 + f_{11}X_1X_2,$$

$$\vec{f} = (f_{00}, f_{01}, f_{10}, f_{11}), \vec{Y} = \big((1 - y_1)(1 - y_2), y_1(1 - y_2), (1 - y_1)y_2, y_1y_2\big).$$

- Write $C = \langle \vec{f}, \vec{G} \rangle + r \cdot H$ and $z = \langle \vec{f}, \vec{Y} \rangle$.
- $\mathcal{R}_{PedPC}(pp) := \left\{ \left((C, z, y_1, \dots, y_\ell), \tilde{f} \right) : \begin{matrix} \tilde{f} \in \mathbb{F}^{\leq 1}[X_1, \dots, X_\ell] \\ z = \langle \vec{f}, \vec{Y} \rangle, C = \langle \vec{f}, \vec{G} \rangle \end{matrix} \right\}$
- We will construct a protocol for Eval with prover complexity $O(N)$, proof size $O(\log N)$ and verifier complexity $O(N)$.

Focus: prove scalar products, $O(\log N)$ communication complexity

Set $r = 0$ initially
P can just send r to V.
Both remove $r \cdot H$ from C

Improve to $\mathrm{polylog}(N)$ later and add ZK

Evaluation protocol overview

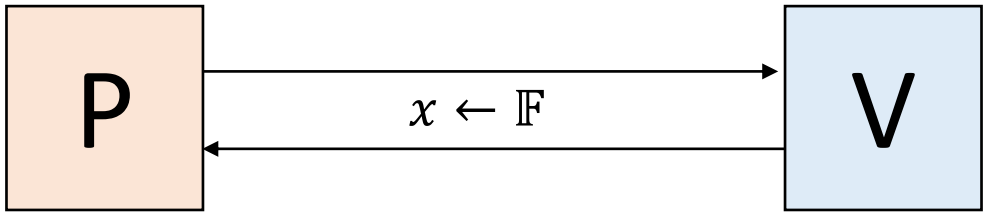
Reduction completeness
⇓
Completeness

Witness:
- vector $\vec{f} \in \mathbb{F}^N$

Length N

Instance:
- commitment $C \in \mathbb{G}$, key $\vec{G} \in \mathbb{G}^N$
- vector $\vec{Y} \in \mathbb{F}^N$, target $z \in \mathbb{F}$

Language:
- $C = \langle \vec{f}, \vec{G} \rangle \in \mathbb{G}$
- $z = \langle \vec{f}, \vec{Y} \rangle \in \mathbb{Z}_p$



No verifier checks!

Length $N/2$

New witness:
- vector $\vec{f}' \in \mathbb{F}^{N/2}$

New instance:
- commitment $C' \in \mathbb{G}$, key $\vec{G}' \in \mathbb{G}^{N/2}$
- vector $\vec{Y}' \in \mathbb{F}^{N/2}$, target $z' \in \mathbb{F}$

New language:
- $C' = \langle \vec{f}', \vec{G}' \rangle \in \mathbb{G}$
- $z' = \langle \vec{f}', \vec{Y}' \rangle \in \mathbb{Z}_p$

$\log N - 1$
⋮ more reductions

Reduction 4-soundness
⇓
$(4, \ldots, 4)$-soundness

Length 1

Final witness:
- element $f^{(\log N)} \in \mathbb{F}$

Final instance:
- commitment $C^{(\log N)} \in \mathbb{G}$, key $G^{(\log N)} \in \mathbb{G}$
- element $Y^{(\log N)} \in \mathbb{F}$, target $z^{(\log N)} \in \mathbb{F}$

Final language:
- $C^{(\log N)} = f^{(\log N)} \cdot G^{(\log N)} \in \mathbb{G}$
- $z^{(\log N)} = f^{(\log N)} \cdot Y^{(\log N)} \in \mathbb{Z}_p$

P simply sends $f^{(\log N)}$ for V to check

Evaluation protocol reduction details

Witness:

- vector $\vec{f} \in \mathbb{F}^N$

Parse

$$\vec{f} = \left(\vec{f}_L, \vec{f}_R\right) \in \mathbb{F}^{N/2} \times \mathbb{F}^{N/2}$$
$$\vec{Y} = \left(\vec{Y}_L, \vec{Y}_R\right) \in \mathbb{F}^{N/2} \times \mathbb{F}^{N/2}$$
$$\vec{G} = \left(\vec{G}_L, \vec{G}_R\right) \in \mathbb{G}^{N/2} \times \mathbb{G}^{N/2}$$

Compute

$C_- = \langle \vec{f}_L, \vec{G}_R \rangle$, $C_+ = \langle \vec{f}_R, \vec{G}_L \rangle$

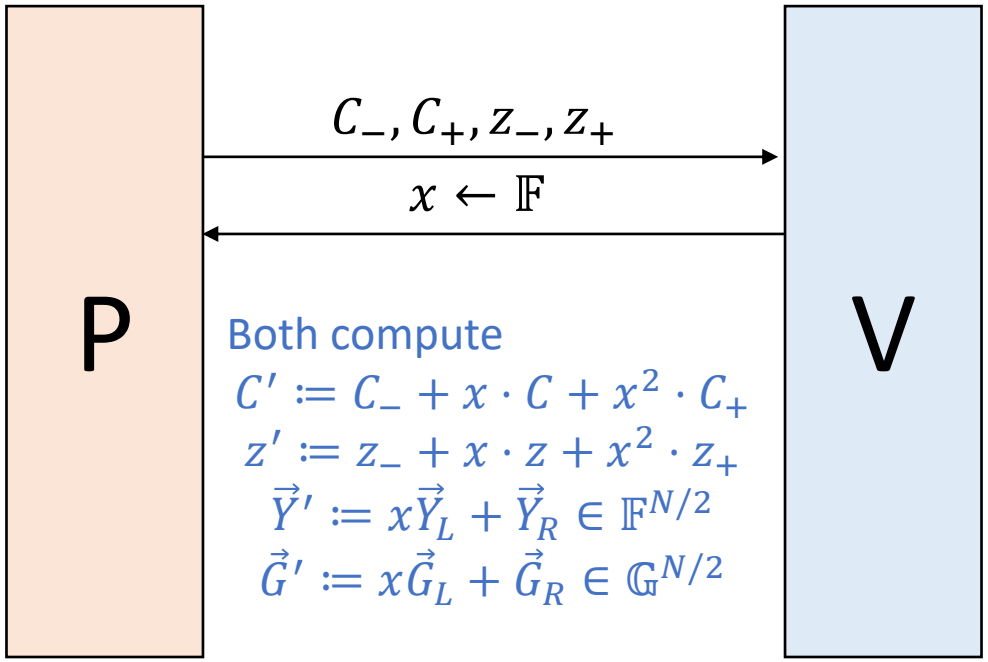
$z_- = \langle \vec{f}_L, \vec{Y}_R \rangle$, $z_+ = \langle \vec{f}_R, \vec{Y}_R \rangle$

Instance:

- commitment $C \in \mathbb{G}$, key $\vec{G} \in \mathbb{G}^N$
- vector $\vec{Y} \in \mathbb{F}^N$, target $z \in \mathbb{F}$

Language:

- $C = \langle \vec{f}, \vec{G} \rangle \in \mathbb{G}$
- $z = \langle \vec{f}, \vec{Y} \rangle \in \mathbb{Z}_p$



$O(\log N)$ communication for full protocol

New witness:

- $\vec{f}' := \vec{f}_L + x\vec{f}_R \in \mathbb{F}^{N/2}$

New instance:

- commitment $C' \in \mathbb{G}$, key $\vec{G}' \in \mathbb{G}^{N/2}$
- vector $\vec{Y}' \in \mathbb{F}^{N/2}$, target $z' \in \mathbb{F}$

New language:

- $C' = \langle \vec{f}', \vec{G}' \rangle \in \mathbb{G}$
- $z' = \langle \vec{f}', \vec{Y}' \rangle \in \mathbb{Z}_p$

Special properties of multicommitments

$$\left\langle \vec{f}_1, \vec{G} \right\rangle + \left\langle \vec{f}_2, \vec{G} \right\rangle = \left\langle \vec{f}_1 + \vec{f}_2, \vec{G} \right\rangle$$

Message homomorphism

Exploit all available structure

$$\left\langle \vec{f}, \vec{G}_1 \right\rangle + \left\langle \vec{f}, \vec{G}_2 \right\rangle = \left\langle \vec{f}, \vec{G}_1 + \vec{G}_2 \right\rangle$$

Bilinearity

Key homomorphism

$$\left\langle \vec{f}_1, \vec{G}_1 \right\rangle + \left\langle \vec{f}_2, \vec{G}_2 \right\rangle = \left\langle \vec{f}_1 || \vec{f}_2, \vec{G}_1 || \vec{G}_2 \right\rangle$$

Concatenation

Completeness analysis of reduction

- We show that valid instance-witness pairs reduce to valid instance-witness pairs i.e. $C' = \langle \vec{f}', \vec{G}' \rangle$ and $z' = \langle \vec{f}', \vec{Y}' \rangle$.

$$\left\langle \vec{f}', \vec{G}' \right\rangle = \left\langle \vec{f}_L + x\vec{f}_R, x\vec{G}_L + \vec{G}_R \right\rangle$$

Expand using bilinearity

$$= \left\langle \vec{f}_L, \vec{G}_R \right\rangle + x\left(\left\langle \vec{f}_L, \vec{G}_L \right\rangle + \left\langle \vec{f}_R, \vec{G}_R \right\rangle\right) + x^2 \left\langle \vec{f}_R, \vec{G}_L \right\rangle$$

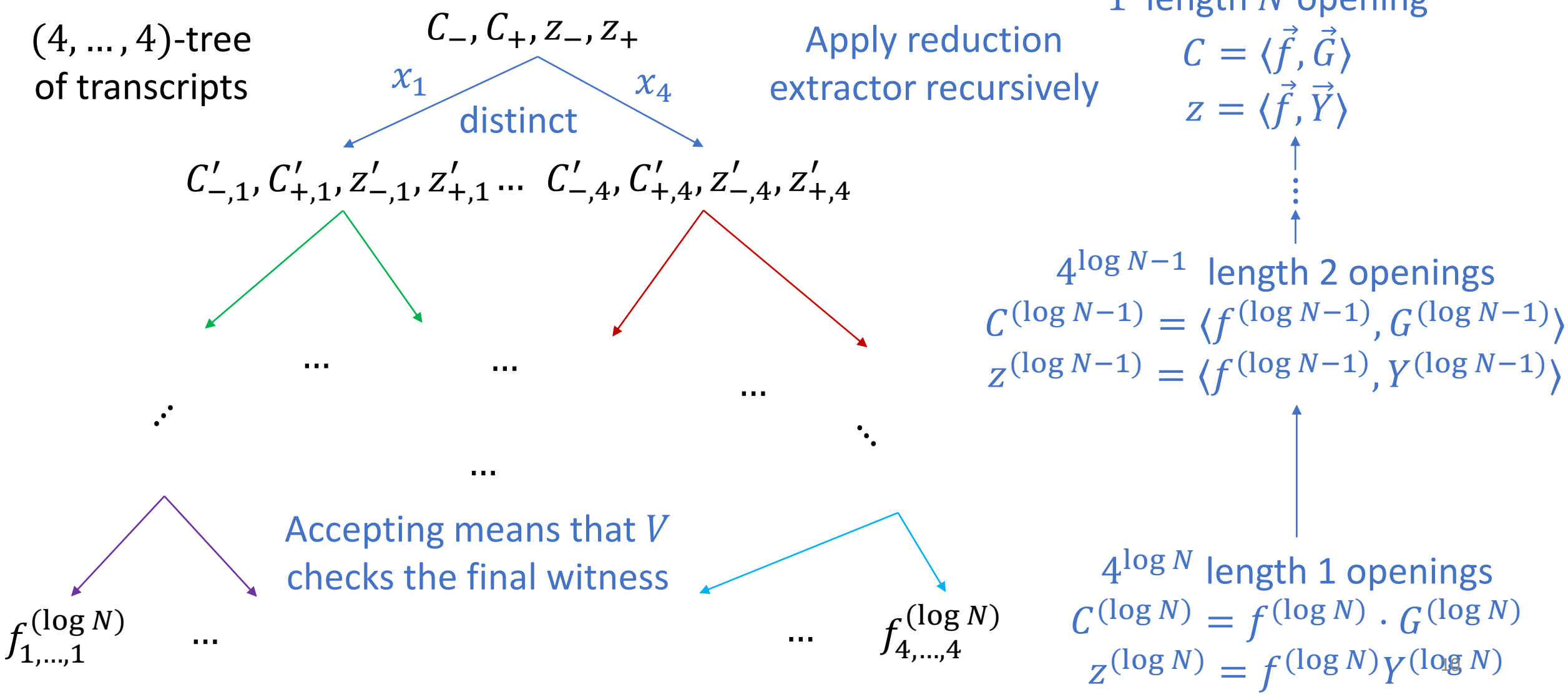
Simplify middle term using concatenation

$$= \left\langle \vec{f}_L, \vec{G}_R \right\rangle + x \left\langle \vec{f}, \vec{G} \right\rangle + x^2 \left\langle \vec{f}_R, \vec{G}_L \right\rangle$$

$$= C_- + x \cdot C + x^2 \cdot C_+ = C'$$

- Similarly, $\left\langle \vec{f}', \vec{Y}' \right\rangle = z_- + x \cdot z + x^2 \cdot z_+ = z'$.

(4,..,4)-soundness from reduction 4-soundness

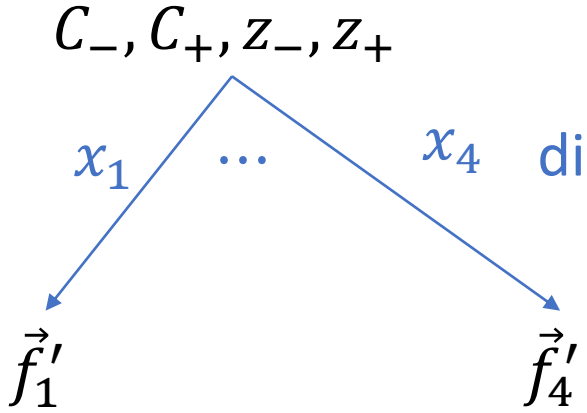


4-soundness of reduction I

$$\vec{Y}_j' \coloneqq x_j \vec{Y}_L + \vec{Y}_R$$
$$\vec{G}_j' \coloneqq x_j \vec{G}_L + \vec{G}_R$$

We cannot make assumptions about how $\vec{f}_j'$ were computed.

- Consider a 4-tree of transcripts.



Satisfying, $\forall j \in \{1,2,3,4\}$,

$$\left\langle \vec{f}_j', \vec{G}_j' \right\rangle = C' = C_- + x_j \cdot C + x_j^2 \cdot C_+$$
$$\left\langle \vec{f}_j', \vec{Y}_j' \right\rangle = z' = z_- + x_j \cdot z + x_j^2 \cdot z_+$$

- We want openings in terms of $\vec{G}$. Bilinearity Concatenation $\vec{f}_{x,j} \coloneqq x_j \vec{f}_j' || \vec{f}_j'$
- $\left\langle \vec{f}_j', \vec{G}_j' \right\rangle = \left\langle \vec{f}_j', x_j \vec{G}_L + \vec{G}_R \right\rangle = x_j \left\langle \vec{f}_j', \vec{G}_L \right\rangle + \left\langle \vec{f}_j', \vec{G}_R \right\rangle = \left\langle x_j \vec{f}_j' || \vec{f}_j', \vec{G}_L || \vec{G}_R \right\rangle = \left\langle \vec{f}_{x,j}, \vec{G} \right\rangle$

Taking $j = 1,2,3$

$$\begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{pmatrix} \begin{pmatrix} C_- \\ C \\ C_+ \end{pmatrix} = \begin{pmatrix} \vec{f}_{x,1}^T \\ \vec{f}_{x,2}^T \\ \vec{f}_{x,3}^T \end{pmatrix} \cdot \begin{pmatrix} G_0 \\ \vdots \\ G_{N-1} \end{pmatrix}$$

Invertibility of Vandermonde matrices

$$\begin{pmatrix} C_- \\ C \\ C_+ \end{pmatrix} = \begin{pmatrix} \vec{f}_- \\ \vec{f} \\ \vec{f}_+ \end{pmatrix} \cdot \begin{pmatrix} G_0 \\ \vdots \\ G_{N-1} \end{pmatrix}$$

4-soundness of reduction II

Extractor output: $\vec{f} \in \mathbb{Z}_p^N$.

$\forall j \in \{1,2,3,4\}$,

$$\left\langle \vec{f}_j', \vec{G}_j' \right\rangle = C' = C_- + x_j \cdot C + x_j^2 \cdot C_+$$

$$\left\langle \vec{f}_j', \vec{Y}_j' \right\rangle = z' = z_- + x_j \cdot z + x_j^2 \cdot z_+$$

Why is the output a witness?

By construction, $C = \langle \vec{f}, \vec{G} \rangle$.

We must show that $z = \langle \vec{f}, \vec{Y} \rangle$ to prove $\vec{f}$ is really a witness.

$$\left\langle \vec{f}_{x,j}, \vec{G} \right\rangle = C_- + x_j \cdot C + x_j^2 \cdot C_+, \qquad \left\langle \vec{f}_{x,j}, \vec{Y} \right\rangle = z_- + x_j \cdot z + x_j^2 \cdot z_+.$$

Similarly to the previous slide

Substituting openings, we have

$$\left\langle \vec{f}_{x,j}, \vec{G} \right\rangle = \left\langle \vec{f}_-, \vec{G} \right\rangle + x_j \cdot \left\langle \vec{f}, \vec{G} \right\rangle + x_j^2 \cdot \left\langle \vec{f}_+, \vec{G} \right\rangle = \left\langle \vec{f}_- + x_j \vec{f} + x_j^2 \vec{f}_+, \vec{G} \right\rangle$$

Hence $\vec{f}_{x,j} = \vec{f}_- + x_j \vec{f} + x_j^2 \vec{f}_+$ or the extractor breaks binding.

Recall that by definition, $\vec{f}_{x,j} \coloneqq x_j \vec{f}_j' || \vec{f}_j'$ so $x_j \vec{f}_j' || \vec{f}_j' = \vec{f}_- + x_j \vec{f} + x_j^2 \vec{f}_+, j \in \{1,2,3,4\}$

4-soundness of reduction III

$x_j \vec{f}_j' || \vec{f}_j' = \vec{f}_- + x_j \vec{f} + x_j^2 \vec{f}_+, j \in \{1,2,3,4\}$

Splitting $\vec{f}_- = \vec{f}_{-,L} || \vec{f}_{-,R}$, $\vec{f} = \vec{f}_L || \vec{f}_R$ and $\vec{f}_+ = \vec{f}_{+,L} || \vec{f}_{+,R}$, we have

$$(x_j \vec{f}_j' || \vec{f}_j') = (\vec{f}_{-,L} || \vec{f}_{-,R}) + x_j (\vec{f}_L || \vec{f}_R) + x_j^2 (\vec{f}_{+,L} || \vec{f}_{+,R})$$

Considering the left and right halves separately,

$$x_j \vec{f}_j' = \vec{f}_{-,L} + x_j \vec{f}_L + x_j^2 \vec{f}_{+,L}, \qquad \vec{f}_j' = \vec{f}_{-,R} + x_j \vec{f}_R + x_j^2 \vec{f}_{+,R}$$

Substituting the right expression for $\vec{f}_j'$ into the left expression,

$$x_j \vec{f}_{-,R} + x_j^2 \vec{f}_R + x_j^3 \vec{f}_{+,R} = \vec{f}_{-,L} + x_j \vec{f}_L + x_j^2 \vec{f}_{+,L}$$

Degree 3, holds for 4 distinct $x_j \Rightarrow$ identically zero

Comparing coefficients,

$$\vec{f}_{-,L} = 0, \qquad \vec{f}_L = \vec{f}_{-,R}, \qquad \vec{f}_R = \vec{f}_{+,L}, \qquad \vec{f}_{+,R} = 0.$$

$$\vec{f}_j' = \vec{f}_L + x_j \vec{f}_R$$

4-soundness of reduction IV

Now we finally show that $z = \langle \vec{f}, \vec{Y} \rangle$.

Recall that $\left\langle \vec{f}_j', \vec{Y}_j' \right\rangle = z_- + x_j \cdot z + x_j^2 \cdot z_+, \quad \forall j \in \{1,2,3,4\},$

$\vec{Y}_j' := x_j \vec{Y}_L + \vec{Y}_R$

Now we know also know

$\vec{f}_j' = \vec{f}_L + x_j \vec{f}_R$

Substituting, we have

$$\left\langle \vec{f}_j', \vec{Y}_j' \right\rangle = \left\langle \vec{f}_L + x_j \vec{f}_R, x_j \vec{Y}_L + \vec{Y}_R \right\rangle$$

Expand using bilinearity

$$= \left\langle \vec{f}_L, \vec{Y}_R \right\rangle + x_j \left(\left\langle \vec{f}_L, \vec{Y}_L \right\rangle + \left\langle \vec{f}_R, \vec{Y}_R \right\rangle \right) + x_j^2 \left\langle \vec{f}_R, \vec{Y}_L \right\rangle$$

Simplify middle term using concatenation

$$= \left\langle \vec{f}_L, \vec{Y}_R \right\rangle + x_j \left\langle \vec{f}, \vec{Y} \right\rangle + x_j^2 \left\langle \vec{f}_R, \vec{Y}_L \right\rangle = z_- + x_j \cdot z + x^2 \cdot z_+$$

Degree 3, holds for 4 distinct $x_j \Rightarrow$ identically zero

Middle term $\Rightarrow \langle \vec{f}, \vec{Y} \rangle = z.$

Communication and prover complexity

Witness:

- vector $\vec{f} \in \mathbb{F}^N$

Compute

$C_- = \langle \vec{f}_L, \vec{G}_R \rangle, C_+ = \langle \vec{f}_R, \vec{G}_L \rangle$

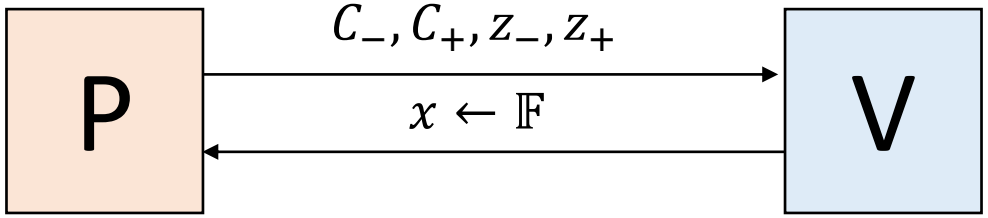
$z_- = \langle \vec{f}_L, \vec{Y}_R \rangle, z_+ = \langle \vec{f}_R, \vec{Y}_R \rangle$

Instance:

- commitment $C \in \mathbb{G}$, key $\vec{G} \in \mathbb{G}^N$
- vector $\vec{Y} \in \mathbb{F}^N$, target $z \in \mathbb{F}$

Language:

- $C = \langle \vec{f}, \vec{G} \rangle \in \mathbb{G}$
- $z = \langle \vec{f}, \vec{Y} \rangle \in \mathbb{Z}_p$



$O(1)$ $\mathbb{F}$ and $\mathbb{G}$ ops in each reduction

Total: $O(\log N)$ ops

Both compute

$C' := C_- + x \cdot C + x^2 \cdot C_+$

$z' := z_- + x \cdot z + x^2 \cdot z_+$

$\vec{Y}' := x\vec{Y}_L + \vec{Y}_R \in \mathbb{F}^{N/2}$

$\vec{G}' := x\vec{G}_L + \vec{G}_R \in \mathbb{G}^{N/2}$

$\log N - 1$ more reductions

$O(N)$ $\mathbb{F}$ and $\mathbb{G}$ ops in first reduction

Total prover complexity:

$O(N + N/2 + \cdots 1) = O(N)$ ops

$O(1)$ communication per round

$O(\log N)$ rounds

$O(\log N)$ communication complexity

Final witness:

- element $f^{(\log N)} \in \mathbb{F}$

Final instance:

- commitment $C^{(\log N)} \in \mathbb{G}$, key $G^{(\log N)} \in \mathbb{G}$
- element $Y^{(\log N)} \in \mathbb{F}$, target $z^{(\log N)} \in \mathbb{F}$

Final language:

- $C^{(\log N)} = f^{(\log N)} \cdot G^{(\log N)} \in \mathbb{G}$
- $z^{(\log N)} = f^{(\log N)} \cdot Y^{(\log N)} \in \mathbb{Z}_p$

P simply sends $f^{(\log N)}$ for V to check

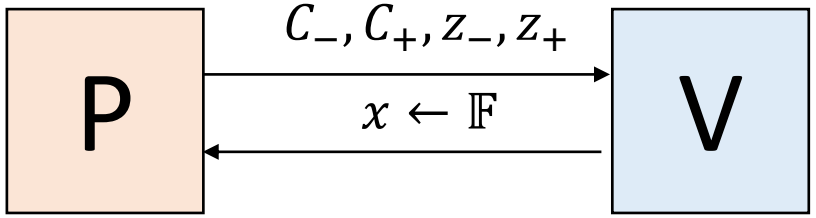
Verifier complexity

Witness:

- vector $\vec{f} \in \mathbb{F}^N$

Instance:

- commitment $C \in \mathbb{G}$, key $\vec{G} \in \mathbb{G}^N$
- vector $\vec{Y} \in \mathbb{F}^N$, target $z \in \mathbb{F}$



$O(1)$ $\mathbb{F}$ and $\mathbb{G}$ ops in each reduction

Total: $O(\log N)$ ops

Both compute

$$C' := C_- + x \cdot C + x^2 \cdot C_+$$
$$z' := z_- + x \cdot z + x^2 \cdot z_+$$
$$\vec{Y}' := x\vec{Y}_L + \vec{Y}_R \in \mathbb{F}^{N/2}$$
$$\vec{G}' := x\vec{G}_L + \vec{G}_R \in \mathbb{G}^{N/2}$$

$O(N)$ $\mathbb{G}$ ops to get $\vec{G}'$

Total verifier complexity for $G^{(\log N)}$:

$O(N + N/2 + \cdots 1) = O(N)$ ops

Compute $Y^{(\log N)}$ all in one at the end

$$\vec{Y} = \left(\widetilde{\mathrm{EQ}}(\vec{y}; \vec{i})\right)_{\vec{i} \in \{0,1\}^{\log N}}$$
$$= \left(\prod_{j=1}^{\log N} \widetilde{\mathrm{EQ}}(y_j; i_j)\right)_{\vec{i} \in \{0,1\}^{\log N}}$$
$$\vec{Y}_L = (1 - y_1) \cdot \left(\prod_{j=2}^{\log N} \widetilde{\mathrm{EQ}}(y_j; i_j)\right)_{\vec{i} \in \{0,1\}^{\log N - 1}}$$
$$\vec{Y}_R = y_1 \cdot \left(\prod_{j=2}^{\log N} \widetilde{\mathrm{EQ}}(y_j; i_j)\right)_{\vec{i} \in \{0,1\}^{\log N - 1}}$$
$$\vec{Y}' =$$
$$(x - xy_1 + y_1)\left(\prod_{j=2}^{\log N} \widetilde{\mathrm{EQ}}(y_j; i_j)\right)_{\vec{i} \in \{0,1\}^{\log N - 1}}$$
$$Y^{(\log N)} = \prod_{j=1}^{\log N} (x_j - x_j y_j + y_j)$$

Costs $O(\log N)$ $\mathbb{F}$-ops

$$G^{(\log N)} = \sum_{i=0}^{N-1} x_1^{i_1} \cdots x_{\log N}^{i_{\log N}} \cdot G_i$$
$$\vec{i} = \mathrm{Binary}(i)$$

Final witness:

- element $f^{(\log N)} \in \mathbb{F}$

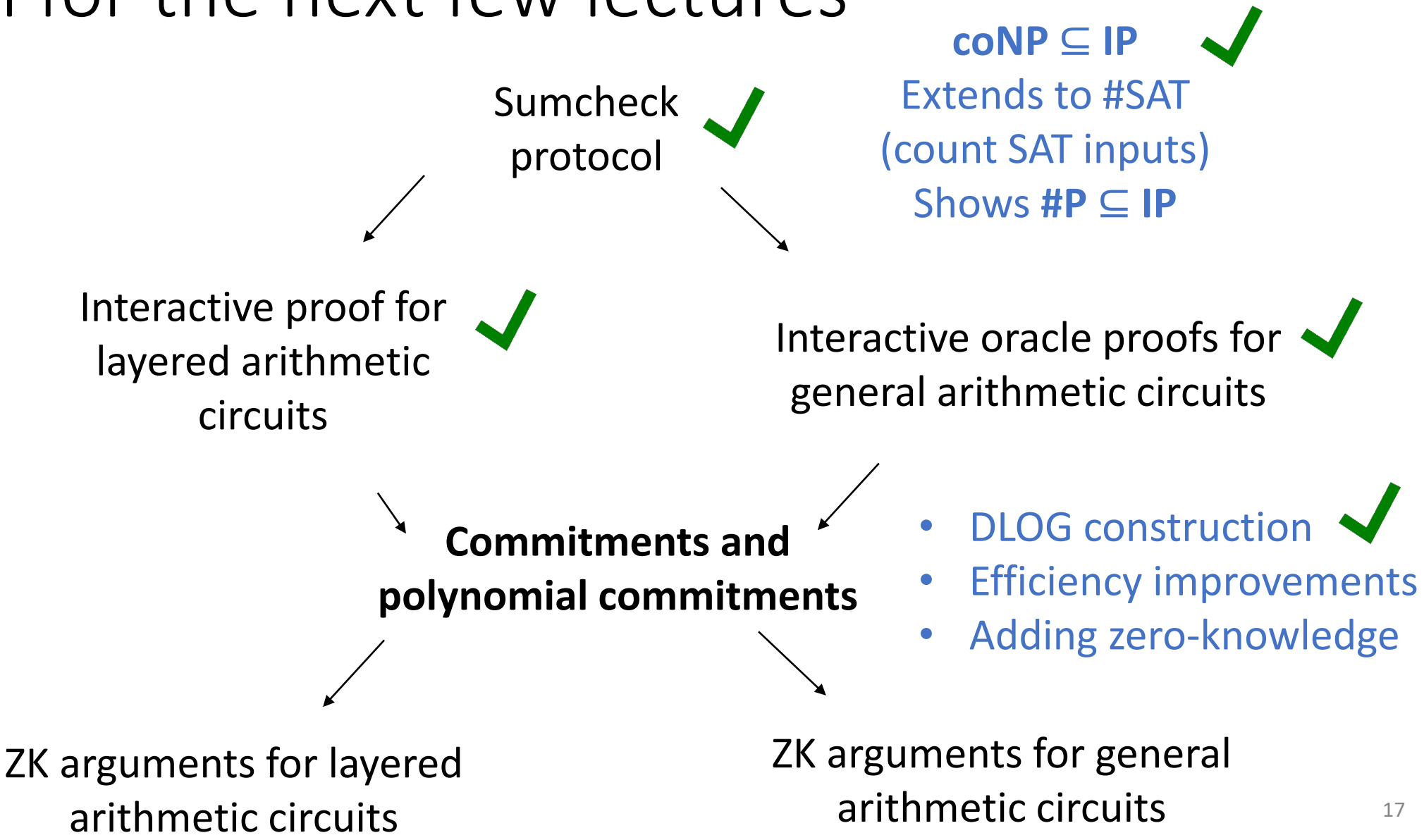
Final instance:

- commitment $C^{(\log N)} \in \mathbb{G}$, key $G^{(\log N)} \in \mathbb{G}$
- element $Y^{(\log N)} \in \mathbb{F}$, target $z^{(\log N)} \in \mathbb{F}$

Final language:

- $C^{(\log N)} = f^{(\log N)} \cdot G^{(\log N)} \in \mathbb{G}$
- $z^{(\log N)} = f^{(\log N)} \cdot Y^{(\log N)} \in \mathbb{Z}_p$

Plan for the next few lectures



Bilinear pairing maps

Note: we will later generalize to cyclic groups of order n when we construct non-interactive proofs

Definition:

A *bilinear group* is a triple of three groups of order p and a *bilinear map* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ satisfying

$$\forall a, b \in \mathbb{Z}_p, \forall G_1 \in \mathbb{G}_1, \forall G_2 \in \mathbb{G}_2,$$
$$e(a \cdot G_1, b \cdot G_2) = ab \cdot e(G_1, G_2)$$

Pairing maps
'multiply DLOGs'

which is non-degenerate i.e.

$$\text{If } \mathbb{G}_1 = \langle G_1 \rangle, \mathbb{G}_2 = \langle G_2 \rangle, \text{ then } \mathbb{G}_T = \langle e(G_1, G_2) \rangle$$

Notation:

'$G_1 \cdot G_2$' means $e(G_1, G_2)$. '$\langle \vec{G}_1, \vec{G}_2 \rangle$' means $\sum_i G_{1,i} \cdot G_{2,i} = \sum_i e(G_{1,i}, G_{2,i})$.

What do $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ look like?

- $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, order p
- There are constructions with

Note: $p \neq t$

$\mathbb{G}_1, \mathbb{G}_2$ elliptic curve groups
$E : Y^2 = X^3 + cX + d \; mod \; t$
for a prime t

Efficiently computable homomorphisms

$\mathbb{G}_T \subseteq \mathbb{F}_{t^k}^{\times}$, multiplicative group
k is called the "embedding degree"

$$k = O(1)$$

- $\mathbb{G}_1 \leftrightarrow \mathbb{G}_2$ "type 1"
- $\mathbb{G}_1 \to \mathbb{G}_2$ "type 2"
- $\mathbb{G}_1 \quad \mathbb{G}_2$ "type 3"

Will use later to construct non-interactive proofs

Will use now

AFGHO Multicommitments

Plain Pedersen:

- $G, H \leftarrow_{\$} \mathbb{G}.$ $\quad m, r \in \mathbb{Z}_p.$
- $C = m \cdot G + r \cdot H \in \mathbb{G}.$
- DLOG: hard to find m, r with $m \cdot G + r \cdot H = 0.$

Plain AFGHO:

- $G, H \leftarrow_{\$} \mathbb{G}_2.$ $\quad M, R \in \mathbb{G}_1.$
- $C = M \cdot G + R \cdot H \in \mathbb{G}_T.$
- DPAIR: hard to find M, R with $M \cdot G + R \cdot H = 0.$

  Implied by DDH in $\mathbb{G}_1, \mathbb{G}_2$

Plain Pedersen:

- $\vec{G}, H$ over $\mathbb{G}.$ $\quad \vec{f}, r$ over $\mathbb{Z}_p.$
- $C = \langle \vec{f}, \vec{G} \rangle + r \cdot H \in \mathbb{G}.$
- Hard to find m, r with $\langle \vec{f}, \vec{G} \rangle + r \cdot H = 0.$

Plain AFGHO:

- $\vec{G}, H$ over $\mathbb{G}_2.$ $\quad \vec{F}, R$ over $\mathbb{G}_1.$
- $C = \langle \vec{F}, \vec{G} \rangle + R \cdot H \in \mathbb{G}_T.$
- Hard to find $\vec{F}, R$ with $\langle \vec{F}, \vec{G} \rangle + R \cdot H = 0.$

Problems and solutions:

- **Problem:** how can we avoid $O(N)$ verifier work in the Eval protocol?
- **Solution:** delegate computation of $G^{(\log N)}$ to P using AFGHO polynomial commitments!
- **Problem:** how can we commit to matrices of size N^2 for the R1CS IOP using commitment keys of size $O(N)$?
- **Solution:** use two-tiered commitments. Commit to the rows of A, B, C using $O(N)$ Pedersen commitments and then commit to the rows using AFGHO commitments.

- The full details take much longer to explain than expected!
- I will explain them in a (fully optional to watch) extra video so that have enough time for the non-interactive ZK section of the course.

Results

A polynomial commitment scheme with

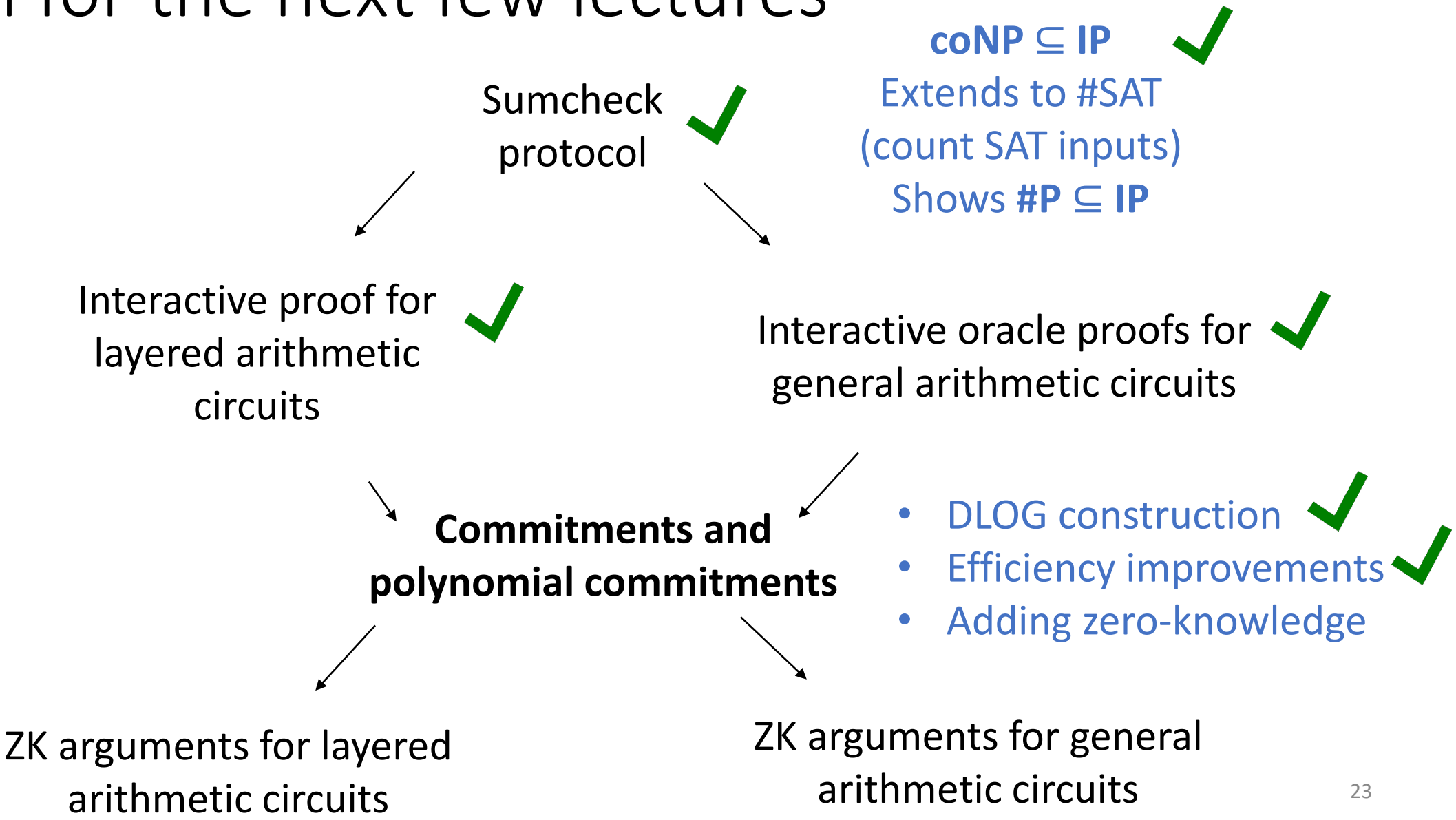
- Public parameters of size $O(N)$

and Eval protocol with

- Preprocessing: $O(N)$ ops
- Prover: $O(N)$ ops for vectors of length N, $O(N + |M|)$ for $N \times N$ matrices
- Verifier: $O\left(\log^2 N\right)$ ops
- Communication: $O\left(\log^2 N\right)$

relying on DDH in $\mathbb{G}_1$ and $\mathbb{G}_2$

Plan for the next few lectures



Hiding polynomial evaluations

Evaluation relation for Pedersen multicommitments:

Given $y_1, \dots, y_{\log N}, z \in \mathbb{F}$, c, prove knowledge of openings $\{f_{\vec{i}}\}, r$ of C such that $\sum_{\vec{i} \in \{0,1\}^{log\ N}} f_{\vec{i}} \cdot \widetilde{\mathrm{EQ}}(\vec{y}; \vec{i}) = z$.

$$\mathcal{R}_{PedPC}(pp) := \left\{ \left((C, z, y_1, \dots, y_\ell), (\tilde{f}, r) \right) : \begin{array}{c} \tilde{f} \in \mathbb{F}^{\leq 1}[X_1, \dots, X_\ell] \\ \tilde{f}(y_1, \dots, y_\ell) = z, C = \langle \vec{f}, \vec{G} \rangle + r \cdot H \end{array} \right\}$$

$z, \vec{y}$ not hidden

Committed evaluation relation for Pedersen multicommitments:

Given $y_1, \dots, y_{\log N}, z \in \mathbb{F}$, c, prove knowledge of openings $\{f_{\vec{i}}\}, r$ of C such that $\sum_{\vec{i} \in \{0,1\}^{log\ N}} f_{\vec{i}} \cdot \widetilde{\mathrm{EQ}}(\vec{y}; \vec{i}) = z$.

pp for Pedersen multicommitments

$$\mathcal{R}_{ComPedPC}(pp, pp') := \left\{ \left((C, C_z, y_1, \dots, y_\ell), (\tilde{f}, r, z, s) \right) : \begin{array}{c} \tilde{f} \in \mathbb{F}^{\leq 1}[X_1, \dots, X_\ell] \\ \tilde{f}(y_1, \dots, y_\ell) = z, C = \langle \vec{f}, \vec{G} \rangle + r \cdot H \\ C_z = z \cdot G + s \cdot H \end{array} \right\}$$

pp' for plain Pedersen commitments

z now hidden

Committed evaluation protocol

Reduction completeness ⇓ Completeness

Reduction 2-soundness ⇓ $(2,4,\dots,4)$-soundness

Witness:

- vector $\vec{f} \in \mathbb{F}^N, r \in \mathbb{F}$
- $z, s \in \mathbb{F}$

Sample $\phi \leftarrow_\$ \mathbb{F}^{\leq 1}[X_1, \dots, X\_\ell]$ with coefficient vector $\vec{\phi} \in \mathbb{F}^N$.
Sample $\rho, \sigma \leftarrow_\$ \mathbb{F}$.
Compute

$$\zeta = \phi(\vec{y})$$
$$D = \left\langle \vec{\phi}, \vec{G} \right\rangle + \rho \cdot H$$
$$D_\zeta = \zeta \cdot G + \sigma \cdot H$$

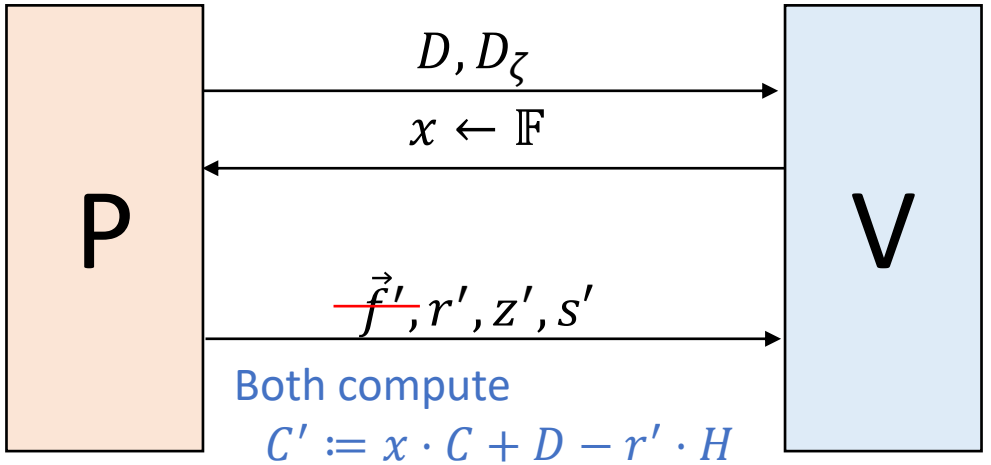
Compute $f' := xf + \phi$
$\vec{f}' := x\vec{f} + \vec{\phi}, z' := xz + \zeta$
$r' := xr + \rho, s' := xs + \sigma$

New witness:

$\vec{f}' := x\vec{f} + \vec{\phi} \in \mathbb{F}^N$

Instance:

- commitment $C \in \mathbb{G}$, key $\vec{G} \in \mathbb{G}^N$
- vector $\vec{Y} \in \mathbb{F}^N$, target $z \in \mathbb{F}$
- commitment $C_z \in \mathbb{G}$, key $G, H \in \mathbb{G}$



New instance:

- commitment $C' \in \mathbb{G}$, key $\vec{G} \in \mathbb{G}^N$
- vector $\vec{Y} \in \mathbb{F}^N$, target $z \in \mathbb{F}$

Run previous Eval protocol

Language:

- $C = \left\langle \vec{f}, \vec{G} \right\rangle + r \cdot H \in \mathbb{G}$
- $z = \left\langle \vec{f}, \vec{Y} \right\rangle \in \mathbb{F}$
- $C_z = z \cdot G + s \cdot H$

Check
$x \cdot C + D == \left\langle \vec{f}', \vec{G} \right\rangle + r' \cdot H$
$f'(\vec{y}) == z'$
Check
$x \cdot C_z + D_\zeta == z' \cdot G + s' \cdot H$

New language:

- $C' = \left\langle \vec{f}', \vec{G} \right\rangle \in \mathbb{G}$
- $z' = \left\langle \vec{f}', \vec{Y} \right\rangle \in \mathbb{F}$

Committed evaluation protocol completeness

- $C = \left\langle \vec{f}, \vec{G} \right\rangle + r \cdot H, z = f(\vec{y}) = \left\langle \vec{f}, \vec{Y} \right\rangle, C_z = z \cdot G + s \cdot H.$ Left-multiply by x
- $D = \left\langle \vec{\phi}, \vec{G} \right\rangle + \rho \cdot H, \zeta = \phi(\vec{y}) = \left\langle \vec{\phi}, \vec{Y} \right\rangle, D_\zeta = \zeta \cdot G + \sigma \cdot H.$

Add

- $C' = \left\langle \vec{f}', \vec{G} \right\rangle + r' \cdot H, z = f(\vec{y}) = \left\langle \vec{f}', \vec{Y} \right\rangle, C'_z = z' \cdot G + s' \cdot H.$
- In the optimized version using Eval, we have reduced to a true instance.
- Completeness follows from the completeness of the previous Eval protocol.

SHVZK analysis

Inefficient protocol

$S(pp, pp', C, C_z, p, x)$

1. $\vec{f}' \leftarrow_{\$} \mathbb{F}^N$. $z' := f(\vec{y})$.
2. $r', s' \leftarrow_{\$} \mathbb{F}$.
3. $D = x \cdot C - \left\langle \vec{f}', \vec{G} \right\rangle - r' \cdot H$.
4. $D_\zeta = x \cdot C_z - z' \cdot G - s' \cdot H$.
5. Output $\left(D, D_\zeta, x, \vec{f}', r', z', s'\right)$.

What is the verifier's view?

- $\left(D, D_\zeta, x, \vec{f}', r', z', s'\right)$ with
- $x \cdot C + D = \left\langle \vec{f}', \vec{G} \right\rangle + r' \cdot H$, $f'(\vec{y}) = z'$ and $x \cdot C_z + D_\zeta = z' \cdot G + s' \cdot H$.
- $\rho \leftarrow_{\$} \mathbb{F}$ so $r' = xr + \rho$ is uniform in $\mathbb{F}$. Similarly for s'.
- $D = x \cdot C - \left\langle \vec{f}', \vec{G} \right\rangle - r' \cdot H$ is uniquely determined. Similarly for D_ζ.

Why is the simulator valid? (efficient, indistinguishable)

- Clearly, the simulator is efficient.
- $\vec{f}', z', r', s'$ have identical distributions to the real protocol.
- The other values are uniquely determined by the verifier checks.

SHVZK analysis

Efficient protocol

Eval protocol transcript

$S(pp, pp', C, C_z, p, x, x_1, \dots, x_\ell)$

1. $\vec{f}' \leftarrow_\$ \mathbb{F}^N$. $z' := f(\vec{y})$. $r', s' \leftarrow_\$ \mathbb{F}$.
2. $D = x \cdot C - \left\langle \vec{f}', \vec{G} \right\rangle - r' \cdot H$
3. $D_\zeta = x \cdot C_z - z' \cdot G - s' \cdot H$
4. Get transcript π by running the honest prover algorithm for Eval on the new instance.
5. Output $(D, D_\zeta, x, r', z', s', \pi)$.

What is the verifier's view?

- $\left(D, D_\zeta, x, \vec{f}', r', z', s', \pi\right)$ with
- $x \cdot C + D = \left\langle \vec{f}', \vec{G} \right\rangle + r' \cdot H$, $f'(\vec{y}) = z'$ and $x \cdot C_z + D_\zeta = z' \cdot G + s' \cdot H$.
- $\rho \leftarrow_\$ \mathbb{F}$ so $r' = xr + \rho$ is uniform in $\mathbb{F}$. Similarly for s'.
- $D = x \cdot C - \left\langle \vec{f}', \vec{G} \right\rangle - r' \cdot H$ is uniquely determined. Similarly for D_ζ.

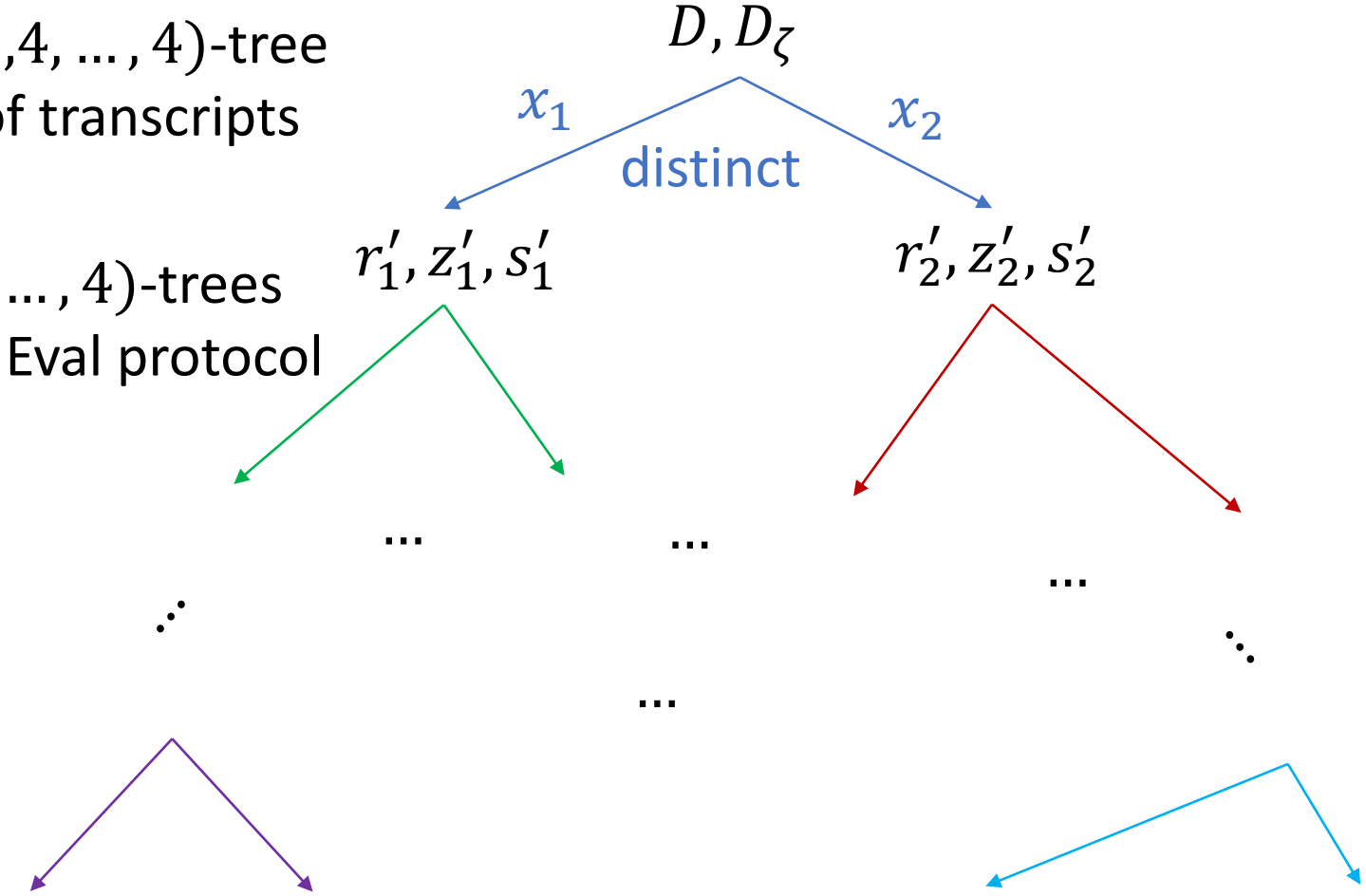
Why is the simulator valid? (efficient, indistinguishable)

- Clearly, the simulator is efficient.
- $\vec{f}', z', r', s'$ have identical distributions to the real protocol.
- The other values are uniquely determined by the verifier checks.
- The Eval prover algorithm is run on the same input distribution in both simulated and real protocol executions.

(2,4,..,4)-soundness from reduction 2-soundness

$(2,4,\ldots,4)$-tree of transcripts

$(4,\ldots,4)$-trees for Eval protocol x2



Witnesses for new instance

$$\langle \vec{f}'_1, \vec{G} \rangle + r'_1 \cdot H = x_1 \cdot C + D$$
$$\langle \vec{f}'_2, \vec{G} \rangle + r'_2 \cdot H = x_2 \cdot C + D$$

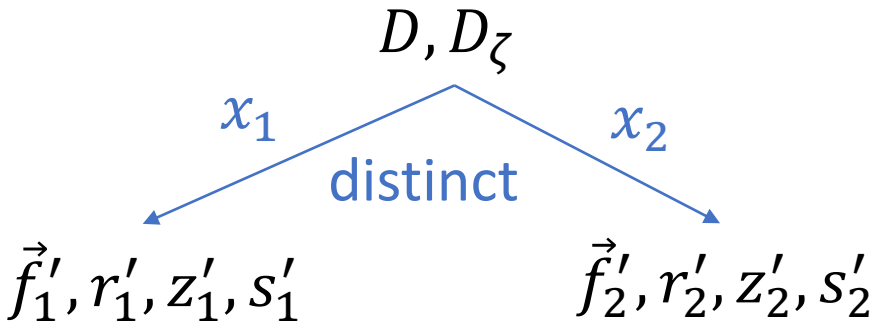
$$C'_1 := x_1 \cdot C + D - r'_1 \cdot H$$
$$C'_2 := x_2 \cdot C + D - r'_2 \cdot H$$

Eval protocol extractor produces $\vec{f}'_1$, $\vec{f}'_2$ such that:

$$C'_1 = \langle \vec{f}'_1, \vec{G} \rangle \qquad C'_2 = \langle \vec{f}'_2, \vec{G} \rangle$$
$$z'_2 = \langle \vec{f}'_1, \vec{Y} \rangle \qquad z'_2 = \langle \vec{f}'_2, \vec{Y} \rangle$$

2-soundness analysis of reduction

2-tree of transcripts



Satisfying

$$\langle \vec{f}_1', \vec{G} \rangle + r_1' \cdot H = x_1 \cdot C + D$$
$$\langle \vec{f}_2', \vec{G} \rangle + r_2' \cdot H = x_2 \cdot C + D$$
$$z_1' = \langle \vec{f}_1', \vec{Y} \rangle$$
$$z_2' = \langle \vec{f}_2', \vec{Y} \rangle$$
$$x_1 \cdot C_z + D_\zeta = z_1' \cdot G + s_1' \cdot H$$
$$x_2 \cdot C_z + D_\zeta = z_2' \cdot G + s_2' \cdot H$$

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix} \begin{pmatrix} D \\ C \end{pmatrix} = \begin{pmatrix} \vec{f}_1' \\ \vec{f}_2' \end{pmatrix} \cdot \vec{G} + \begin{pmatrix} r_1' \\ r_2' \end{pmatrix} \cdot H$$

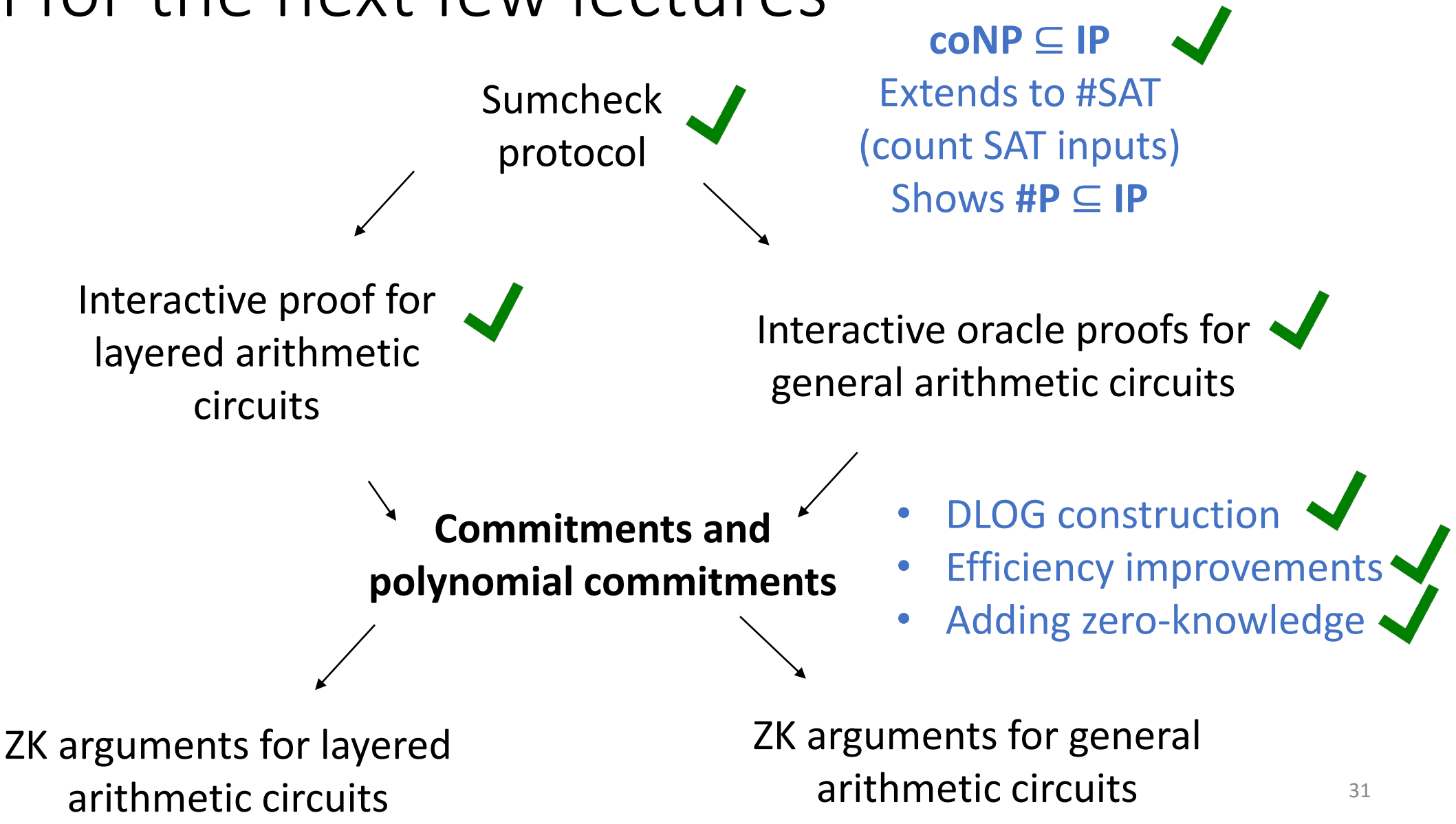
$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \end{pmatrix} \begin{pmatrix} D_\zeta \\ C_z \end{pmatrix} = \begin{pmatrix} z_1' \\ z_2' \end{pmatrix} \cdot G + \begin{pmatrix} s_1' \\ s_2' \end{pmatrix} \cdot H = \begin{pmatrix} \vec{f}_1' \\ \vec{f}_2' \end{pmatrix} \cdot (\vec{Y} \cdot G) + \begin{pmatrix} s_1' \\ s_2' \end{pmatrix} \cdot H$$

Inverting the linear system, we get $\vec{f}, \vec{\phi}, r, s, \rho, \sigma$ satisfying

$$\begin{pmatrix} D \\ C \end{pmatrix} = \begin{pmatrix} \vec{\phi} \\ \vec{f} \end{pmatrix} \cdot \vec{G} + \begin{pmatrix} \rho \\ r \end{pmatrix} \cdot H,$$

$$\begin{pmatrix} D_\zeta \\ C_z \end{pmatrix} = \begin{pmatrix} \vec{\phi} \\ \vec{f} \end{pmatrix} \cdot (\vec{Y} \cdot G) + \begin{pmatrix} \sigma \\ s \end{pmatrix} \cdot H = \begin{pmatrix} \langle \vec{\phi}, \vec{Y} \rangle \\ \langle \vec{f}, \vec{Y} \rangle \end{pmatrix} \cdot G + \begin{pmatrix} \sigma \\ s \end{pmatrix} \cdot H$$
$$= f(\vec{y})$$

Plan for the next few lectures



How to make protocols succinct and ZK

Succinctness:

- Replace each large/oracle message with a polynomial commitment.
- Whenever V would have made a polynomial evaluation query to perform a check, P sends the evaluation, and they run the Eval protocol together.

Zero-knowledge:

- Also commit to each small message with a plain Pedersen commitment.
- Whenever V would have made a polynomial evaluation query to perform a check, P sends a commitment to the evaluation, and they run the hidden Eval protocol together.
- P and V run Σ-protocols on committed values to check that each verification equation would have been satisfied.

Compiling GKR to a ZK argument

- GKR language: $\left\{\left(\{add_i, mult_i\}_{i=0}^{D-1}, \vec{x}, \vec{y}\right) : C(\vec{x}) = \vec{y}\right\}$. **P** language

- Compiled GKR relation:

Easily generalizes to different lengths

$$\left\{\left(\left(\{add_i, mult_i\}_{i=0}^{D-1}, C_{\vec{x}}, C_{\vec{y}}\right), (\vec{x}, r, \vec{y}, s)\right) : \begin{array}{c} C_{\vec{x}} = \langle \vec{x}, \vec{G} \rangle + r \cdot H \\ C_{\vec{y}} = \langle \vec{y}, \vec{G} \rangle + s \cdot H \\ C(\vec{x}) = \vec{y} \end{array}\right\}.$$

Instance: $\left(\{add_i, mult_i\}_{i=0}^{D-1}, C_{\vec{x}}, C_{\vec{y}}\right)$ — (polynomial) commitments

Witness: $(\vec{x}, r, \vec{y}, s)$

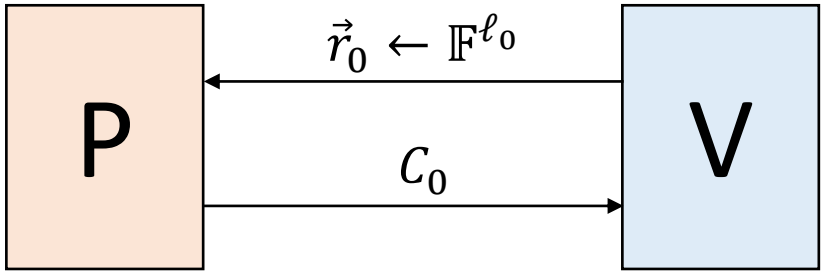
NP relation

Example: initial reduction

w_0 computed from $\vec{x}$

Initial claim: $C_{\vec{y}}$ contains $\vec{y}$ satisfying $\widetilde{w}_0 \equiv \tilde{y}$

Initial claim: $\widetilde{w}_0 \equiv \tilde{y}$



Compute $v_0 := \tilde{y}(\vec{r}_0)$.
Sample $s \leftarrow_{\$} \mathbb{F}$.
Compute $C_0 = v_0 \cdot G + s \cdot H$.

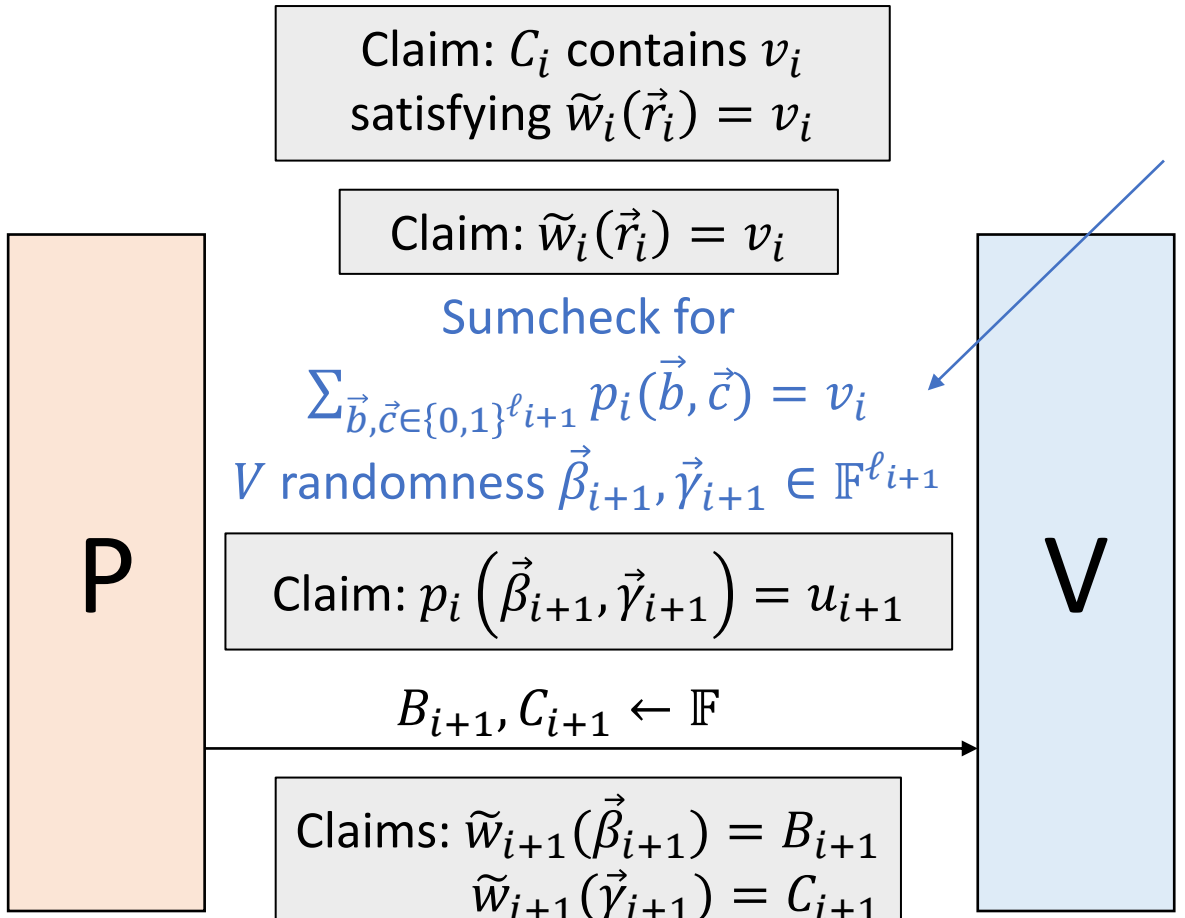
Both compute $v_0 := \tilde{y}(\vec{r}_0)$

Claim: $\widetilde{w}_0(\vec{r}_0) = v_0$

Claim: C_0 contains v_0 satisfying $\widetilde{w}_0(\vec{r}_0) = v_0$

Hidden eval protocol proves: $C_{\vec{y}}, C_0$ contain $\vec{y}, v_0$ satisfying $\tilde{y}(\vec{r}_0) = v_0$

Example: sumcheck level i to level i+1 reduction



- P sends *commitments* to sumcheck messages.
- V uses Σ-protocols to perform sumcheck checks but on committed values.
- V computes commitment U_{i+1} to final target evaluation using homomorphism.

Sample $r_i, s_i \leftarrow \mathbb{F}$.

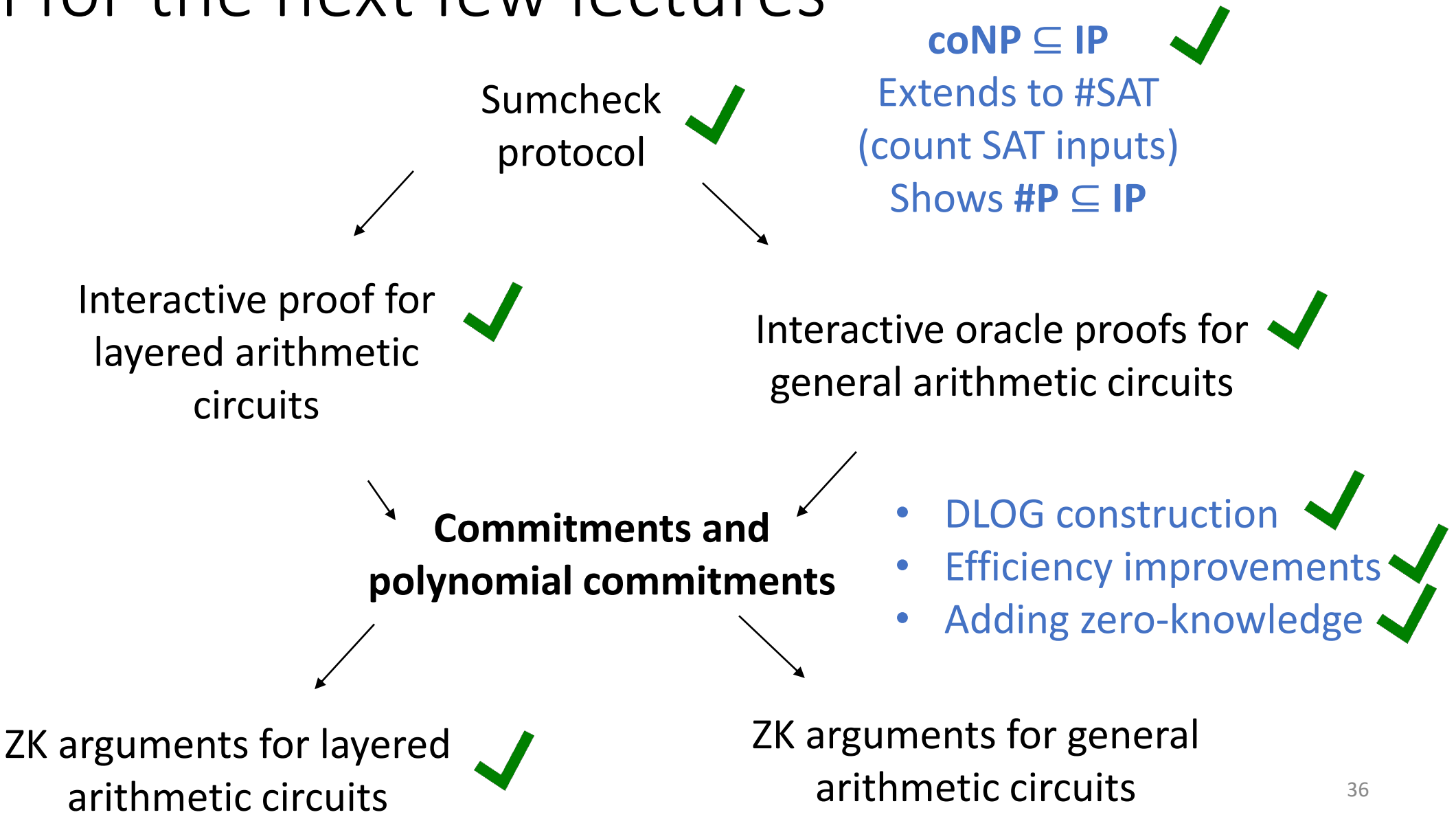
$B_{i+1} := \widetilde{w}_{i+1}(\vec{\beta}_{i+1})$
$C_{i+1} := \widetilde{w}_{i+1}(\vec{\gamma}_{i+1})$

$D_{i+1} = B_{i+1} \cdot G + r_i \cdot H$
$E_{i+1} = C_{i+1} \cdot G + s_i \cdot H$

Both P, V run a Σ-protocol to check that $D_{i+1}, E_{i+1}, U_{i+1}$ contain $B_{i+1}, C_{i+1}, u_{i+1}$ satisfying

$$\widetilde{\mathrm{add}}_i\left(\vec{r}_i, \vec{\beta}_{i+1}, \vec{\gamma}_{i+1}\right)[B_{i+1} + C_{i+1}] + \widetilde{\mathrm{mult}}_i\left(\vec{r}_i, \vec{\beta}_{i+1}, \vec{\gamma}_{i+1}\right)[B_{i+1} \cdot C_{i+1}] == u_{i+1}$$

Plan for the next few lectures



Compiling R1CS IOP to a ZK argument

$$\mathcal{R}_{R1CS} = \left\{ \left((\mathbb{F}, A, B, C, \vec{x}) , \vec{w} \right) : \begin{array}{c} A, B, C \in \mathbb{F}^{N_r \times N_c}, \vec{x} \in \mathbb{F}^k \\ \vec{w} \in \mathbb{F}^{N_c - k}, \vec{z} := \vec{x} || \vec{w} \\ A\vec{z} \circ B\vec{z} = C\vec{z} \end{array} \right\}.$$

NP relation

- Compiled R1CS relation:

$$\left\{ \left((\mathbb{F}, A, B, C, \vec{x}, C_{\vec{w}}) , (\vec{w}, r) \right) : \begin{array}{c} A, B, C \in \mathbb{F}^{N \times N}, \vec{x} \in \mathbb{F}^k \\ C_{\vec{w}} = \langle \vec{w}, \vec{G} \rangle + r \cdot H \\ \vec{w} \in \mathbb{F}^{N-k}, \vec{z} := \vec{x} || \vec{w} \\ A\vec{z} \circ B\vec{z} = C\vec{z} \end{array} \right\}.$$

Instance

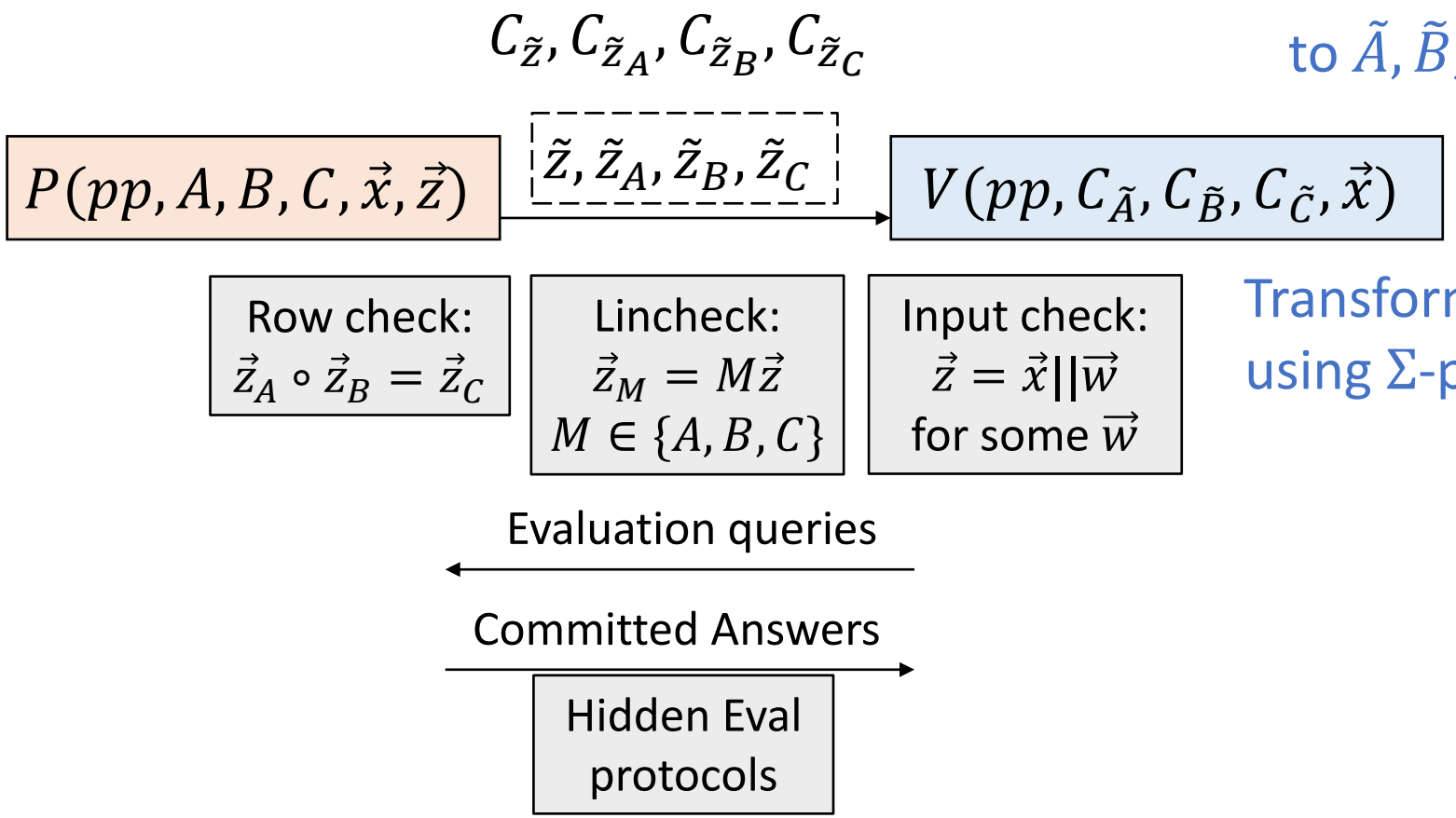
Witness

Alternatively, send $C_{\vec{w}}$ in first message

NP relation

Preprocessing ZK argument for R1CS

Preprocessing computes commitments $C_{\tilde{A}}, C_{\tilde{B}}, C_{\tilde{C}}$ to $\tilde{A}, \tilde{B}, \tilde{C}$



Transform verifier checks using Σ-protocols as with GKR

Security sketch

Completeness:

- Inherited from the underlying IP/IOP, hidden Eval and Σ-protocols

SHVZK:

- Sample uniformly random commitments C and C_z for each polynomial f and evaluation z.
- Sample uniformly random commitments for each small message.
- Simulate the Σ-protocols and hidden Eval protocols.

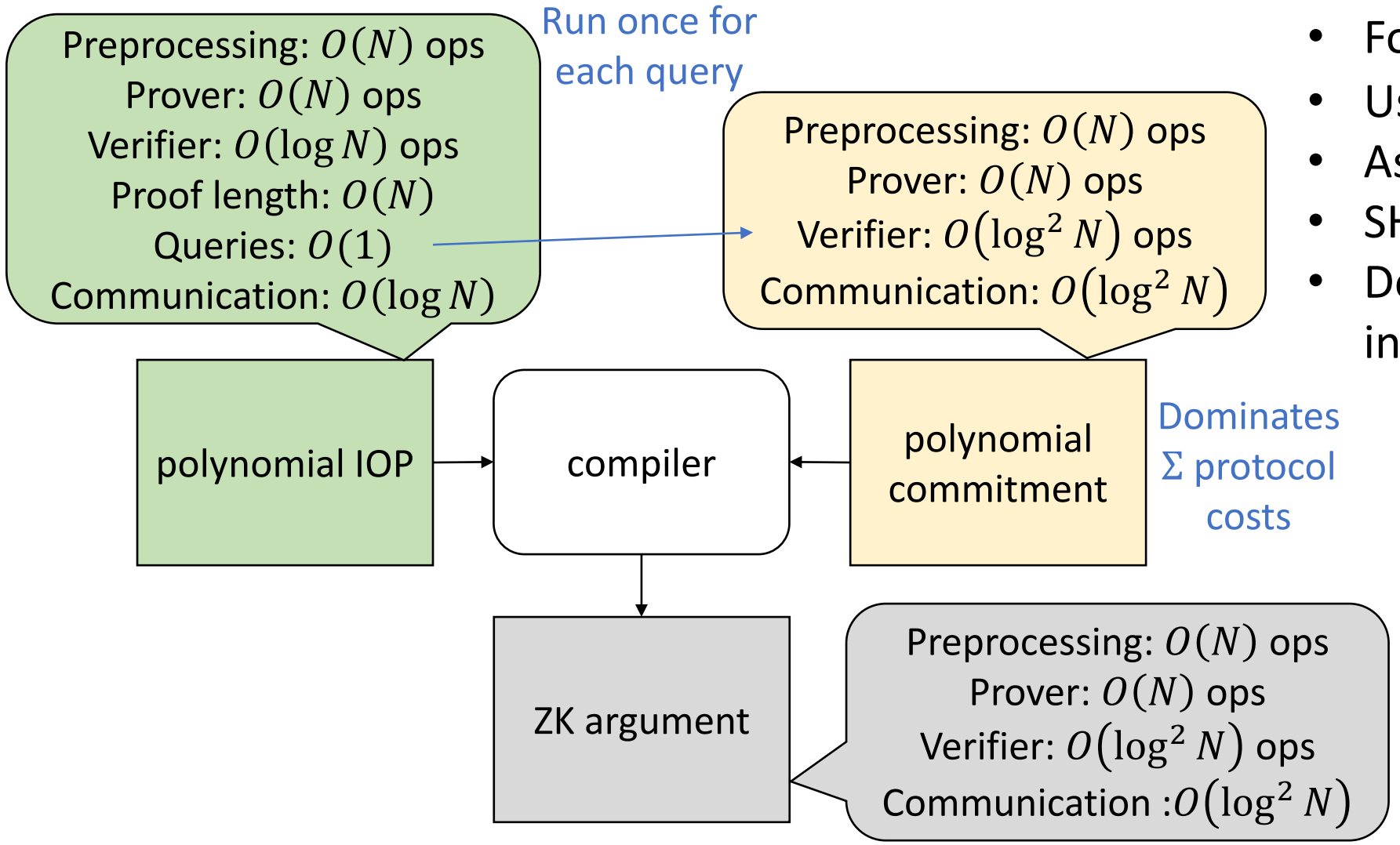
Soundness:

ZK argument adversary → (Hidden eval, Σ-protocol extractors) → IP/IOP argument adversary

Knowledge error bounds in terms of

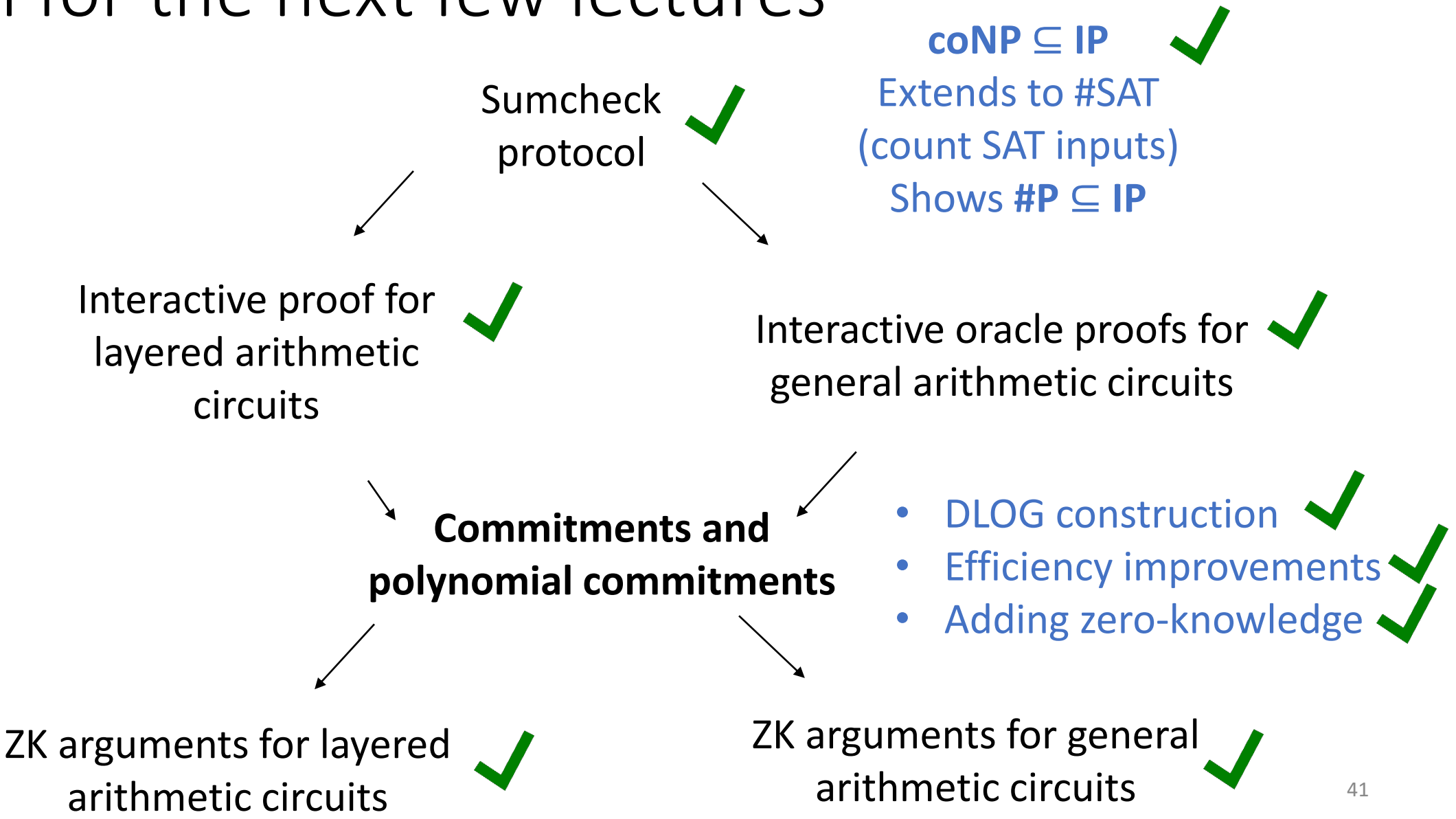
- IOP/IP soundness error
- Hidden eval knowledge error
- Σ-protocol knowledge error

Summary of CSAT argument result



- For CSAT over $\mathbb{Z}_p$.
- Uses $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$
- Assumes DDH in $\mathbb{G}_1$, $\mathbb{G}_2$.
- SHVZK.
- Deterministic, circuit-independent preprocessing.

Plan for the next few lectures



ETH Zurich, Department of Computer Science — Dr. Jonathan Bootle
HS 2023 — Karen Klein, Varun Maram, Antonio Merino Gallardo

Zero-Knowledge Proofs
Exercise 10

10.1 Sumcheck for Multilinear Polynomials with Streaming Log-Space Prover

Consider the sumcheck protocol for multi-linear polynomials on ℓ variables over a field $\mathbb{F}$ (which have $n := 2^\ell$ coefficients in $\mathbb{F}$, indexed by binary strings of length ℓ) and summation domain $\{0,1\}$, where the coefficients of such polynomials are in the *Lagrange basis* (see Lecture 8 slides). Suppose that instead of the full list of the polynomial coefficients, the prover is given access to an oracle that it can query to obtain the next coefficient in the sequence (ordering by binary strings in ascending order). The prover can make as many passes over the sequence as it wishes.

a) Show that the prover can complete the protocol using $O(\log(n)\log(|\mathbb{F}|))$ bits of memory.

b) What is the running time of the prover using this method? How many passes over the sequence does the prover require?

10.2 Analysis of GKR Protocol

Consider a circuit that consists of a tree of multiplication gates on $n = 2^d$ inputs. Address the following tasks.

a) Recall the $mult_i$ maps from the GKR protocol presented in the lectures. Write expressions for $\widetilde{mult}_i$ (their multi-linear extensions) that allow to evaluate them in $O(\log n)$ time.

b) What are the computational costs for the prover and verifier in the GKR protocol?

10.3 R1CS

a) Show that R1CS satisfiability over a field $\mathbb{F}$ is NP-complete.

HINT: Show for any circuit satisfiability instance over a field $\mathbb{F}$, there is an R1CS instance which is satisfiable if and only if the circuit is satisfiable.

b) Let p be a prime number and $\ell := \lfloor \log_2 p \rfloor - 1$. Find an R1CS instance with public instance vector $x = (1, \bar{a}, \bar{b}) \in \mathbb{Z}_p^3$ which is satisfied if and only if $a \leq b$, where a and b are ℓ-bit integer representations of $\bar{a}$ and $\bar{b}$ respectively (in particular, we have $\bar{a}, \bar{b} < 2^\ell$).

c) A point on an elliptic curve over a finite field $\mathbb{Z}_p$ can be represented as a pair $(x_0, y_0) \in \mathbb{Z}_p^2$ satisfying $y_0^2 = x_0^3 + ax_0 + b$.

Consider the "point-doubling" function $[2]\colon \mathbb{Z}_p^2 \to \mathbb{Z}_p^2$ which maps curve points to new curve points, computed as

Function $[2](x_0, y_0)$**:**

1. Compute $t_0 := 3x_0^2 + a$.

2. Compute $u_0 := t_0/(2y_0)$.
3. Compute $x_1 := u_0^2 - 2x_0$.
4. Compute $y_1 := u_0(x_0 - x_1) - y_0$.
5. **Return** $(x_1, y_1) \in \mathbb{Z}_p^2$.

Let $\mathbf{z} = (1, a, x_0, y_0, t_0, u_0, x_1, y_1) \in \mathbb{Z}_p^8$.
Describe matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{Z}_p^{4\times 8}$ such that $\mathbf{Az} \circ \mathbf{Bz} = \mathbf{Cz}$ if and only if $(x_1, y_1) = [2](x_0, y_0)$. You may assume that $y_0 \neq 0$ so that Step 2 in the algorithm for [2] is always well defined.

ETH Zurich, Department of Computer Science — Dr. Jonathan Bootle
HS 2023 — Karen Klein, Varun Maram, Antonio Merino Gallardo

Zero-Knowledge Proofs
Exercise 10

10.1 Sumcheck for Multilinear Polynomials with Streaming Log-Space Prover

Consider the sumcheck protocol for multi-linear polynomials on ℓ variables over a field $\mathbb{F}$ (which have $n := 2^\ell$ coefficients in $\mathbb{F}$, indexed by binary strings of length ℓ) and summation domain $\{0,1\}$, where the coefficients of such polynomials are in the *Lagrange basis* (see Lecture 8 slides). Suppose that instead of the full list of the polynomial coefficients, the prover is given access to an oracle that it can query to obtain the next coefficient in the sequence (ordering by binary strings in ascending order). The prover can make as many passes over the sequence as it wishes.

a) Show that the prover can complete the protocol using $O(\log(n)\log(|\mathbb{F}|))$ bits of memory.

Solution: It suffices for the prover to compute and send the evaluations $q_i(0)$ and $q_i(1)$ in the i-th round. Since f is multi-linear, it can be written as

$$\sum_{s\in\{0,1\}^\ell} f(s)\prod_{j=1}^{\ell} s_jX_j + (1-s_j)(1-X_j),$$

where we use the notational convention that $a^0 = 1$ even when $a = 0 \in \mathbb{F}$. (Note that here we are considering coefficients in the "Lagrange basis", in contrast to coefficients in the "monomial basis" – i.e., $\sum_{s\in\{0,1\}^\ell} a_s \prod_{j=1}^{\ell} X_j^{s_j}$. However the analysis below can be extended to the latter case to obtain a prover space complexity of $O(\log(n)\log(|\mathbb{F}|))$ bits.)

Therefore, for $b_{i+1},\dots,b_\ell \in \{0,1\}$,

$$\begin{aligned}
&f(r_1,\dots,r_{i-1},0,b_{i+1},\dots,b_\ell)\\
&= \sum_{s\in\{0,1\}^\ell} f(s)\Big(\prod_{j=1}^{i-1} s_jr_j + (1-s_j)(1-r_j)\Big)\cdot(1-s_i)\cdot\Big(\prod_{j=i+1}^{\ell} s_jb_j + (1-s_j)(1-b_j)\Big)\\
&= \sum_{s\in\{0,1\}^{i-1}} f(s,0,b_{i+1},\dots,b_\ell)\Big(\prod_{j=1}^{i-1} s_jr_j + (1-s_j)(1-r_j)\Big),
\end{aligned}$$

and similarly for $f(r_1,\dots,r_{i-1},1,b_{i+1},\dots,b_\ell)$. It follows that

$$q_i(0) = \sum_{s\in\{0,1\}^{i-1}} \Big(\prod_{j=1}^{i-1} s_jr_j + (1-s_j)(1-r_j)\Big) \sum_{b_{i+1},\dots,b_\ell\in\{0,1\}} f(s,0,b_{i+1},\dots,b_\ell)$$

and

$$q_i(1) = \sum_{s\in\{0,1\}^{i-1}} \Big(\prod_{j=1}^{i-1} s_jr_j + (1-s_j)(1-r_j)\Big) \sum_{b_{i+1},\dots,b_\ell\in\{0,1\}} f(s,1,b_{i+1},\dots,b_\ell).$$

For any $s \in \{0,1\}^{i-1}$, the prover can compute $\prod_{j=1}^{i-1} s_j r_j + (1-s_j)(1-r_j)$ on its own by only storing the random values $r_1, \ldots, r_{i-1}$. Besides, for any $s \in \{0,1\}^{i-1}$, the prover can compute the sum $\sum_{b_{i+1},\ldots,b_\ell \in \{0,1\}} f(s, 0, b_{i+1}, \ldots, b_\ell)$ in constant space by querying each coefficient $f(s, 0, b_{i+1}, \ldots, b_\ell)$ one at a time for increasing values of $b_{i+1}, \ldots, b_\ell$, and adding the returned value to the partial sum so far. Similarly for the sum $\sum_{b_{i+1},\ldots,b_\ell \in \{0,1\}} f(s, 1, b_{i+1}, \ldots, b_\ell)$. Throughout the protocol, the prover thus needs to store at most ℓ random values and $O(1)$ variables containing fields elements, hence the claims.

b) What is the running time of the prover using this method? How many passes over the sequence does the prover require?

Solution: At each round of the sumcheck protocol, the prover must compute at most 2^ℓ products of at most ℓ field elements and then the sum of 2^ℓ field elements. That is, at most $O(n \log n)$ field operations per round of the protocol, so the prover has an overall complexity of order $O\left(n \log^2 n\right)$. The prover requires one pass over the sequence of coefficients per round of the sumcheck protocol, i.e., $\log n$ passes.

10.2 Analysis of GKR Protocol

Consider a circuit that consists of a tree of multiplication gates on $n = 2^d$ inputs. Address the following tasks.

a) Recall the $mult_i$ maps from the GKR protocol presented in the lectures. Write expressions for $\widetilde{mult_i}$ (their multi-linear extensions) that allow to evaluate them in $O(\log n)$ time.

Solution: Consider a binary tree of multiplication gates. For $i \leq d-1$, let $mult_i \colon \{0,1\}^i \times \{0,1\}^{i+1} \times \{0,1\}^{i+1} \to \{0,1\}$. Define wire labels such that $mult_i(a,b,c) = 1$ if $b = (a,0)$ and $c = (a,1)$, and $mult_i(a,b,c) = 0$ otherwise. Let $a = (a_1, \ldots, a_i)$, $b = (b_1, \ldots, b_{i+1})$ and $c = (c_1, \ldots, c_{i+1})$. For $j \leq i$, the polynomial $(1-a_j)(1-b_j)(1-c_j) + a_j b_j c_j$ evaluates to 1 on $\{0,1\}^3$ if and only if $a_j = b_j = c_j$. The polynomial $1 - b_{i+1}$ evaluates to 1 on $\{0,1\}$ if and only if $b_{i+1} = 0$ and c_{i+1} evaluates to 1 on $\{0,1\}$ if and only if $c_{i+1} = 1$.

This means that we can write

$$\widetilde{mult}_i(a,b,c) = (1-b_{i+1}) \cdot c_{i+1} \cdot \prod_{j=1}^{i} \left[(1-a_j)(1-b_j)(1-c_j) + a_j b_j c_j\right] \ .$$

Each of the i terms in the product can be evaluated in $O(1)$ operations, and since $i \leq d-1$, the entire polynomial can be evaluated in $O(d) = O(\log n)$ operations.

b) What are the computational costs for the prover and verifier in the GKR protocol?

Solution: In the case of a binary tree of multiplication gates, let $r \in \mathbb{F}^i$ and let $p(x,y) := \widetilde{mult}_i(r,x,y)\left[\tilde{w}_{i+1}(x) \cdot \tilde{w}_{i+1}(y)\right]$ be the polynomial used in the sumcheck protocol in the i-th step of the GKR protocol, using formal variables $x = (x_1, \ldots, x_{i+1})$ and $y = (y_1, \ldots, y_{i+1})$.

In the GKR protocol, the sumcheck protocol is used to check whether $\tilde{w}_i(r) = \sum_{b,c \in \{0,1\}^{i+1}} p(b,c)$. Let $\beta = (\beta_1, \ldots, \beta_{i+1})$, $\gamma = (\gamma_1, \ldots, \gamma_{i+1})$ be the sequence of random challenges used in the sumcheck protocol. In the j-th step of the sumcheck protocol, the prover must compute the polynomial

$$q_j(x_j) = \sum_{b \in \{0,1\}^{i+1-j}, c \in \{0,1\}^{i+1}} \widetilde{mult}_i(r, \beta[1:j-1], x_j, b, c)\left[\tilde{w}_{i+1}(\beta[1:j-1], x_j, b) \cdot \tilde{w}_{i+1}(c)\right]$$

$$= \sum_{c \in \{0,1\}^{i+1}} \widetilde{mult}_i(r, \beta[1:j-1], x_j, (c[j+1:i], 0), c)\left[\tilde{w}_{i+1}(\beta[1:j-1], x_j, (c[j+1:i], 0)) \cdot \tilde{w}_{i+1}(c)\right]$$

if $1 \leq j \leq i$, because the $\widetilde{mult_i}$ map defined in task a) above evaluates to 1 only if the bits of b and c agree as $b[1:i-j] = c[j+1:i]$ and $b_{i-j+1} = 0$, for bit-strings $b \in \{0,1\}^{1+1-j}$ and $c \in \{0,1\}^{i+1}$.

Similarly, the prover computes the polynomial

$$q_j(y_{j'}) = \sum_{c \in \{0,1\}^{i+1-j'}} \widetilde{mult}_i(r, \beta, \gamma[1:j'-1], y_{j'}, c) \left[\tilde{w}_{i+1}(\beta) \cdot \tilde{w}_{i+1}(\gamma[1:j'-1], y_{j'}, c)\right]$$

for $j' = j - i$ if $i + 1 \leq j \leq 2i$.

To compute the values of "$\tilde{w}_{i+1}(\beta[1:j], x_j, b)$" above, we can use "book-keeping tables" as follows: the prover first stores in a table A, at respective positions b', the values $\tilde{w}_{i+1}(b')$ for all $b' \in \{0,1\}^{i+1}$; the prover knows all initial wire values of the circuit at setup time (or can be pre-computed from the input values in $O(n)$ time). The prover looks-up the $\tilde{w}_{i+1}(\cdot)$ values from this table A when computing $q_1(x_1)$ in the first step of the sumcheck protocol. Then upon receiving the random value β_1, the prover updates the table entries at all positions $b'' \in \{0,1\}^i$ using the interpolation $A[b''] \leftarrow A[b''](1-\beta_1) + A\left[b'' + 2^i\right]\beta_1$, effectively storing the value $\tilde{w}_{i+1}(\beta_1, b'')$ – which the prover can then use in the second step of the sumcheck protocol when computing $q_2(x_2)$. Note that the table entries $A[b'' + 2^i]$ for $b'' \in \{0,1\}^i$ are now unnecessary to complete the rest of the protocol, i.e., the size of the relevant part of the table has been halved. Since the prover can proceed similarly in the subsequent j steps of the sumcheck protocol ($1 \leq j \leq i$) with the table size halving each time, the above observations suggest that the prover takes about $\sum_{j=1}^{i} O\left(2^{i-j}\right) = O\left(2^i\right)$ operations in total across all i rounds of the sumcheck protocol to compute the values of $\tilde{w}_{i+1}(\beta[1:j], x_j, b)$. The prover does the same "lookup-and-update" procedure to compute the values of "$\tilde{w}_{i+1}(\gamma[1:j], y_j, c)$".

Now given the requisite evaluations of $\tilde{w}_i$, it costs $O(\log n)$ operations to compute an evaluation of $\widetilde{mult}_i$ for each of the $O(2^i)$ terms in the sum for $q_j(x_j)$. This cost is multiplied by $O(2^i)$ operations to compute each term in the sum and then add them together, giving a cost of $O(2^i \log n)$ for a single round. The i-th sumcheck protocol has $2i = O(\log n)$ rounds. The cost of computing polynomials in the first i rounds dominates the cost of the later rounds, so the total cost of the i-th sumcheck protocol is $O(2^i \log^2 n)$. Summing over i gives a total cost of $O(n \log^2 n)$ for the GKR protocol.

The cost for the verifier is dominated by the cost of evaluating $\tilde{w}_d$, which is $O(n)$ operations.

10.3 R1CS

a) Show that R1CS satisfiability over a field $\mathbb{F}$ is NP-complete.

HINT: Show for any circuit satisfiability instance over a field $\mathbb{F}$, there is an R1CS instance which is satisfiable if and only if the circuit is satisfiable.

Solution: Consider a CSAT instance C over a field. Each addition $x + y = z$ can be seen as

$$\begin{bmatrix}0 & 1 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\x\\y\\z\end{bmatrix} \circ \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\x\\y\\z\end{bmatrix} = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\x\\y\\z\end{bmatrix}$$

and each multiplication $xy = z$ as

$$\begin{bmatrix}0 & 1 & 0 & 0\end{bmatrix}\begin{bmatrix}1\\x\\y\\z\end{bmatrix} \circ \begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}1\\x\\y\\z\end{bmatrix} = \begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\x\\y\\z\end{bmatrix}.$$

On the other hand, given public matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, the equation $\mathbf{Az} \circ \mathbf{Bz} = \mathbf{Cz}$ is satisfied if and only if $\langle \mathbf{A}_i, \mathbf{z}\rangle\langle \mathbf{B}_i, \mathbf{z}\rangle = \langle \mathbf{C}_i, \mathbf{z}\rangle$ for all $i \in [\![n]\!]$ if n denotes the number of rows of the matrices. Define then matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ with one row per circuit gate such that the "sub-row" induced by the two inputs and the output of the gate are as above depending on whether the gate is an addition or a multiplication gate. As a result, the circuit is then satisfiable if and only if the R1CS instance is.

b) Let p be a prime number and $\ell \coloneqq \lfloor \log_2 p \rfloor - 1$. Find an R1CS instance with public instance vector $x = (1, \bar{a}, \bar{b}) \in \mathbb{Z}_p^3$ which is satisfied if and only if $a \leq b$, where a and b are ℓ-bit integer representations of $\bar{a}$ and $\bar{b}$ respectively (in particular, we have $\bar{a}, \bar{b} < 2^\ell$).

Solution: Denote by $\mathbf{0}$ and $\mathbf{1}$ the (row) vectors in $\mathbb{Z}_p^\ell$ with respectively all zeroes and all ones, and let $\mathbf{2}^\ell \coloneqq \begin{bmatrix}1 & 2 & 2^2 & \cdots & 2^{\ell-1}\end{bmatrix}$. Also, let $-\mathbf{I}_\ell$ be a matrix which is like the identity matrix $\mathbf{I}_\ell \in \mathbb{Z}_p^{\ell\times\ell}$ but where the "1" entries are replaced by "-1".

Given $b \geq a$, let $(w_i)_{i=0}^{\ell-1} \in \{0,1\}^\ell$ be the binary decomposition of $b - a$, i.e., $b - a = \sum_{i=0}^{\ell-1} w_i 2^i$. The inequality $b \geq a$ then holds if and only if there exists $\mathbf{w} \in \mathbb{Z}_p^\ell$ such that $\mathbf{w} \circ (\mathbf{1} - \mathbf{w}) = \mathbf{0} \bmod p$ and $\langle \mathbf{w}, \mathbf{2}^\ell\rangle = \bar{b} - \bar{a} \bmod p$; note that if $b < a$, then we have $\langle \mathbf{w}, \mathbf{2}^\ell\rangle \neq \bar{b} - \bar{a} \bmod p$ conditional on $\mathbf{w} \circ (\mathbf{1} - \mathbf{w}) = \mathbf{0} \bmod p$ since $\langle \mathbf{w}, \mathbf{2}^\ell\rangle < 2^\ell < p/2$ whereas $\bar{b} - \bar{a} \bmod p = p - (a - b) > p/2$.

It then suffices to define the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{Z}_p^{(\ell+1)\times(\ell+3)}$ as

$$\mathbf{A} = \begin{bmatrix}\mathbf{0}_{1\times 3} & \mathbf{2}^\ell\\ \mathbf{0}_{\ell\times 3} & \mathbf{I}_\ell\end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix}1 & & & \mathbf{0}\\ 1 & & & \\ \vdots & & \mathbf{0}_{(\ell+1)\times 2} & -\mathbf{I}_\ell\\ 1 & & & \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix}0 & -1 & 1 & \mathbf{0}\\ & \mathbf{0}_{\ell\times(\ell+3)} & & \end{bmatrix}.$$

The first row ensures that $\langle \mathbf{w}, \mathbf{2}^{\ell+1}\rangle = \bar{b} - \bar{a} \bmod p$ and the next ℓ rows that $w_i \in \{0,1\}$ for $i \in [\![0, \ell - 1]\!]$.

c) A point on an elliptic curve over a finite field $\mathbb{Z}_p$ can be represented as a pair $(x_0, y_0) \in \mathbb{Z}_p^2$ satisfying $y_0^2 = x_0^3 + ax_0 + b$.

Consider the "point-doubling" function $[2]\colon \mathbb{Z}_p^2 \to \mathbb{Z}_p^2$ which maps curve points to new curve points, computed as

Function $[2](x_0, y_0)$**:**

1. Compute $t_0 := 3x_0^2 + a$.
2. Compute $u_0 := t_0/(2y_0)$.
3. Compute $x_1 := u_0^2 - 2x_0$.
4. Compute $y_1 := u_0(x_0 - x_1) - y_0$.
5. **Return** $(x_1, y_1) \in \mathbb{Z}_p^2$.

Let $\mathbf{z} = (1, a, x_0, y_0, t_0, u_0, x_1, y_1) \in \mathbb{Z}_p^8$.
Describe matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{Z}_p^{4\times 8}$ such that $\mathbf{Az} \circ \mathbf{Bz} = \mathbf{Cz}$ if and only if $(x_1, y_1) = [2](x_0, y_0)$. You may assume that $y_0 \neq 0$ so that Step 2 in the algorithm for [2] is always well defined.
Solution: We define matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{Z}_p^{4\times 8}$ as follows:

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The first row guarantees $t_0 = 3x_0^2 + a$, or equivalently $t_0 - a = 3x_0^2$. The second row guarantees $u_0 = t_0/(2y_0)$, since this is equivalent to $2y_0u_0 = t_0$. Similarly, the third row guarantees $x_1 = u_0^2 - 2x_0$, and the fourth row $y_1 = u_0(x_0 - x_1) - y_0$.