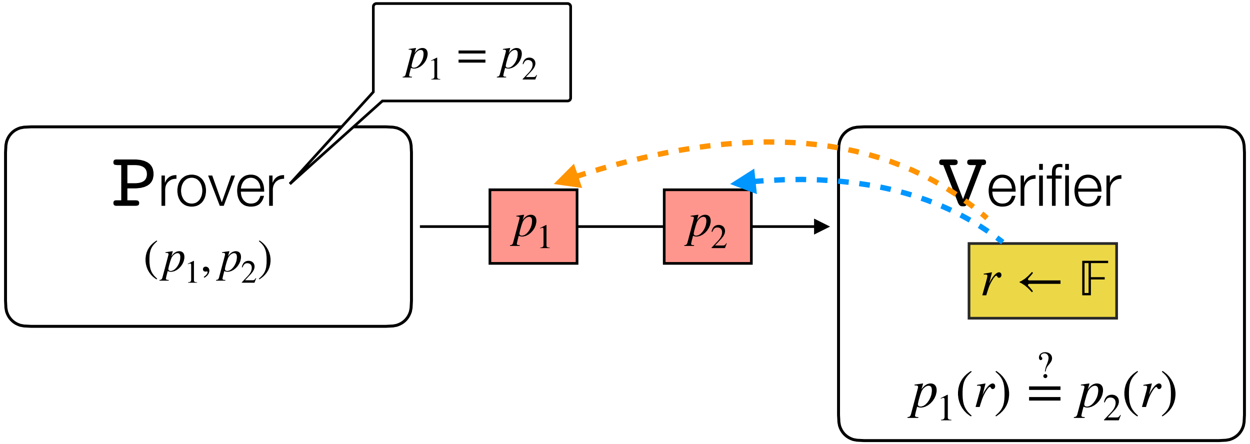
Succinct Arguments

Lecture 04:
PIOP for R1CS

Pratyush Mishra
UPenn
Fall 2025

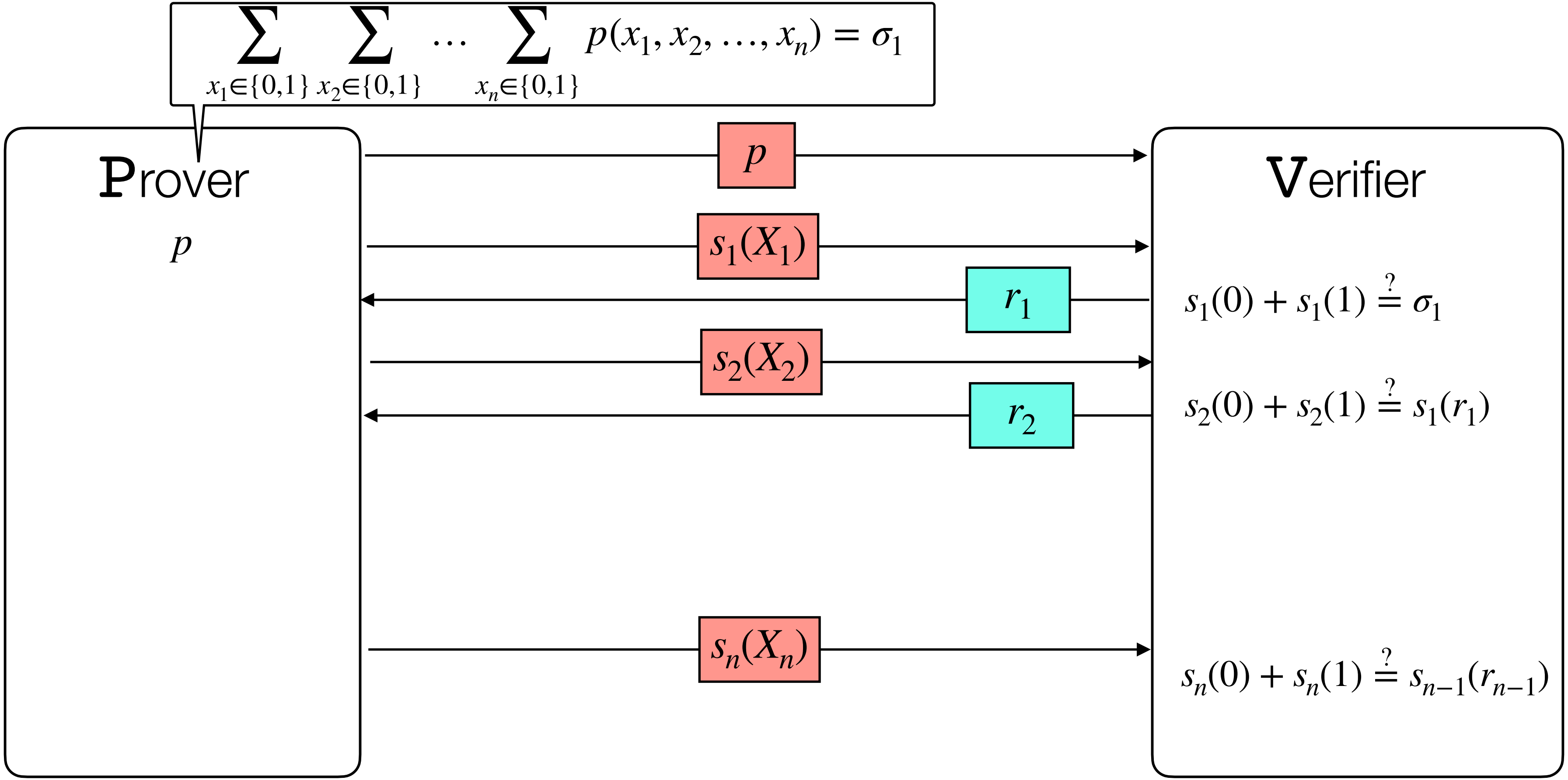
A toolkit of PIOPs

Warmup: PIOP for Equality (Schwartz-Zippel Lemma)



- **Completeness**: If $p_1 = p_2$, then definitely $p_1(r) = p_2(r)$.
- **Soundness**: If $p_1 \neq p_2$, then $p_1(r) = p_2(r) \implies r$ is a root of $q := p_1 - p_2$. But since r is random, this happens with probability $\frac{\deg(q)}{|\mathbb{F}|}$
- Generalizes to multilinear/multivariate polynomials.

Sumcheck protocol



Multivariate Zerocheck [LFKN90]

- Input: V given oracle access to a n-variate polynomial p over field $\mathbb{F}$ and claimed sum $\sigma = \sigma_1$.
- Goal: check the claim:

$$\forall b_1, b_2, \ldots, b_n \in \{0,1\}, \quad p(b_1, \ldots, b_n) = 0$$

Zerocheck Protocol

- **Obervation**: $\forall b_1, b_2, \ldots, b_n \in \{0,1\}, \;\; p(b_1, \ldots, b_n) = 0$ iff $q(X) = \sum_i p(\vec{i}) \cdot X^i = 0$, where $\vec{i}$ is binary decomposition of i.
- Idea: Simply evaluate $q(X)$ at a random point r!
- But how to do evaluation? Naively, would have to query all points of p!
- Idea: sumcheck! $q(r) = \sum_i p(\vec{i}) \cdot r^i = 0$ is a sum check claim!
- Problem: $(1, r, r^2, \ldots)$ is not a polynomial, but a function!
- Idea: interpolate into polynomial! Let $\tilde{r}(X_1, \ldots, X_n)$ be interpolation over hypercube
- At the end of the sumcheck protocol, verifier needs to evaluate p and $\tilde{r}$ at random point. How to evaluate the latter?

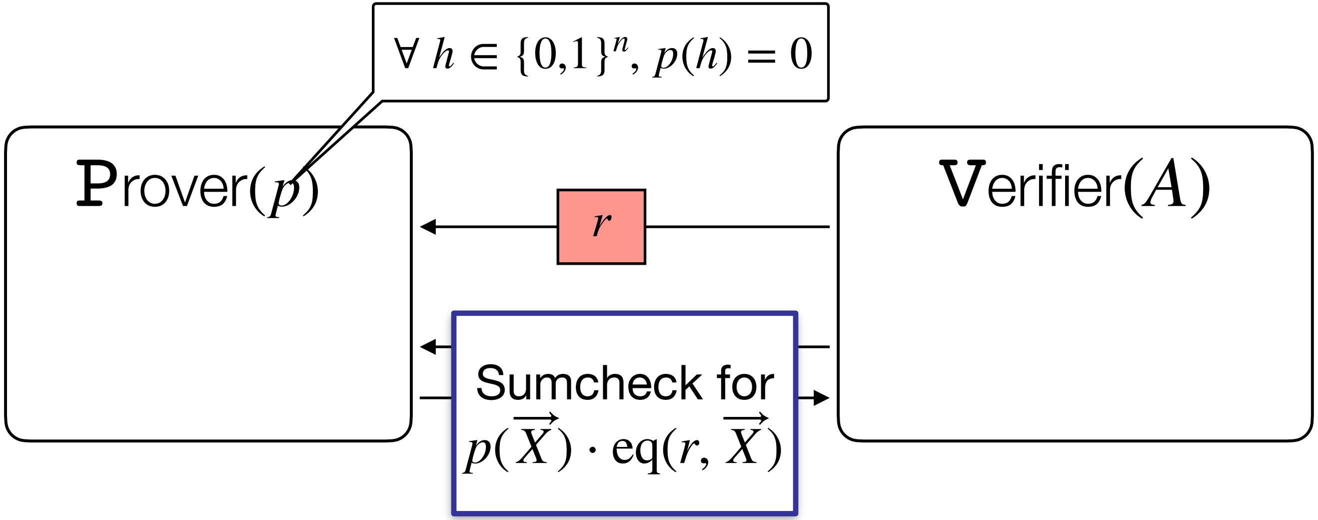
Zerocheck Protocol

- **Obervation: Use multilinear polynomials instead of univariate!**
- **We want *multilinear* q *such that*** $\forall b_1, b_2, \ldots, b_n \in \{0,1\}, \;\; p(b_1, \ldots, b_n) = 0$ iff

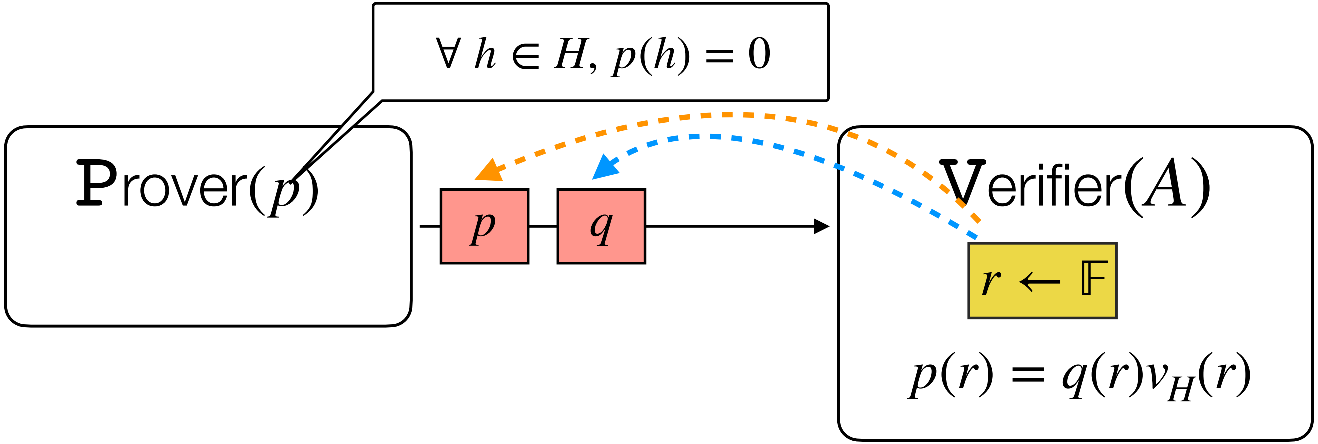
$$q(X_1, \ldots, X_n) = \sum_i p(\vec{i}) \cdot ??? = 0$$

- What to put in ???
- For univariate we used powers of X; what can we use for multilinear?
- Lagrange basis polynomials, ie $\mathrm{eq}(i, X_1, \ldots, X_n)$!

Multilinear ZeroCheck



Univariate ZeroCheck



Lemma: $\forall h \in H,\ \ p(h) = 0$ if and only if $\exists q$ such that $p = q \cdot v_H$.

- **Completeness**: Follows from lemma, and completeness of previous PIOP.
- **Soundness**: The lemma means that we have to check only equality of polynomials via the previous PIOP, and so soundness reduces to that of the previous PIOP.

Lemma: univariate sum check

$$\sum_{h \in H} p(h) = \sigma$$

$$\Longleftrightarrow$$

$\exists\ g$ s.t. $p(X) - (X \cdot g(X) + \frac{\sigma}{|H|}) = 0$ over H

A PIOP for R1CS

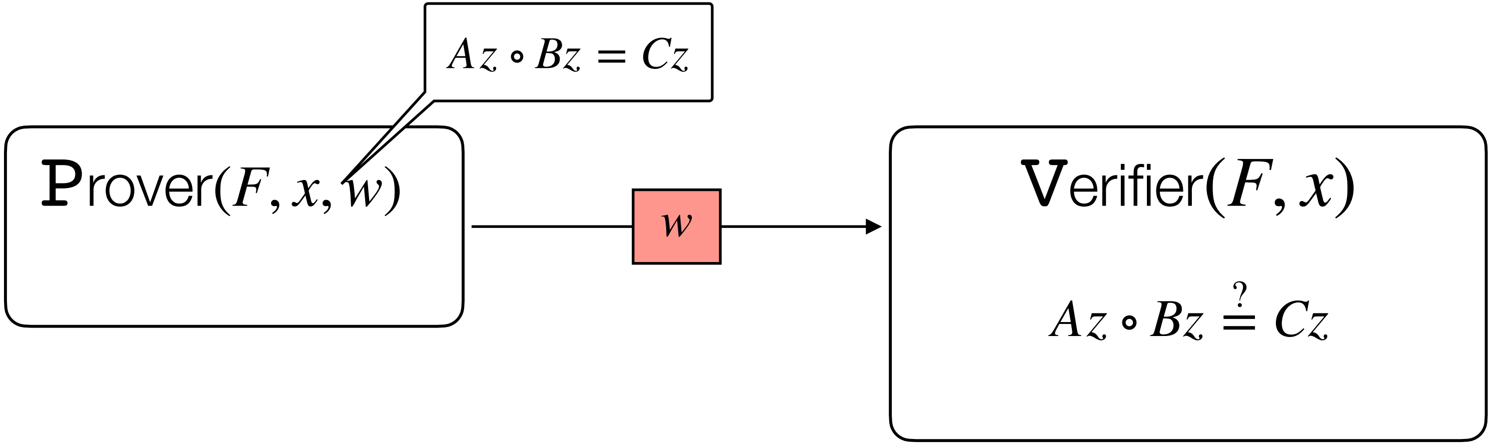
R1CS

An rank-1 constraint system (R1CS) is a generalization of arithmetic circuits

$$(F := (\mathbb{F}, n \in \mathbb{N}, A, B, C), x, w)$$

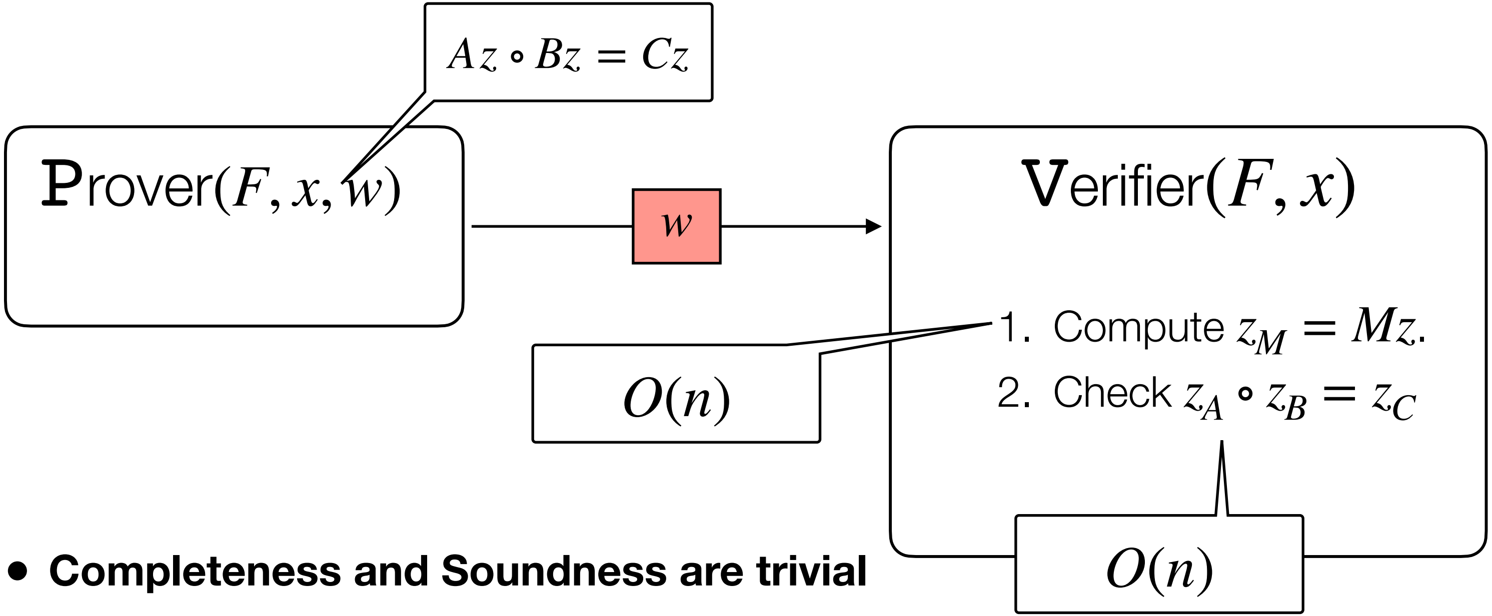
$$z := \begin{bmatrix} x \\ w \end{bmatrix} \qquad \begin{bmatrix} A \end{bmatrix} \begin{bmatrix} z \end{bmatrix} \circ \begin{bmatrix} B \end{bmatrix} \begin{bmatrix} z \end{bmatrix} = \begin{bmatrix} C \end{bmatrix} \begin{bmatrix} z \end{bmatrix}$$

Strawman 1



- **Completeness and Soundness are trivial**
- **What about efficiency?**

Strawman 1



- **Completeness and Soundness are trivial**
- **What about efficiency?**

What checks do we need?

Step 1: Correct Hadamard product

check that for each i, $z_A[i] \cdot z_B[i] = z_C[i]$

Step 2: Correct matrix multiplication

check that $Mz = z_M \quad \forall M \in \{A, B, C\}$

PIOP for Hadamard Product

Prover(F, x, w)

1. Let $H \subseteq \mathbb{F}$ be a set of size n.
2. Interpolate z_A, z_B, z_C to get p_A, p_B, p_C.
3. Run PIOP for zerocheck for polynomial $p_A \cdot p_B - p_C$.

p_A p_B p_C →

Verifier(F, x)

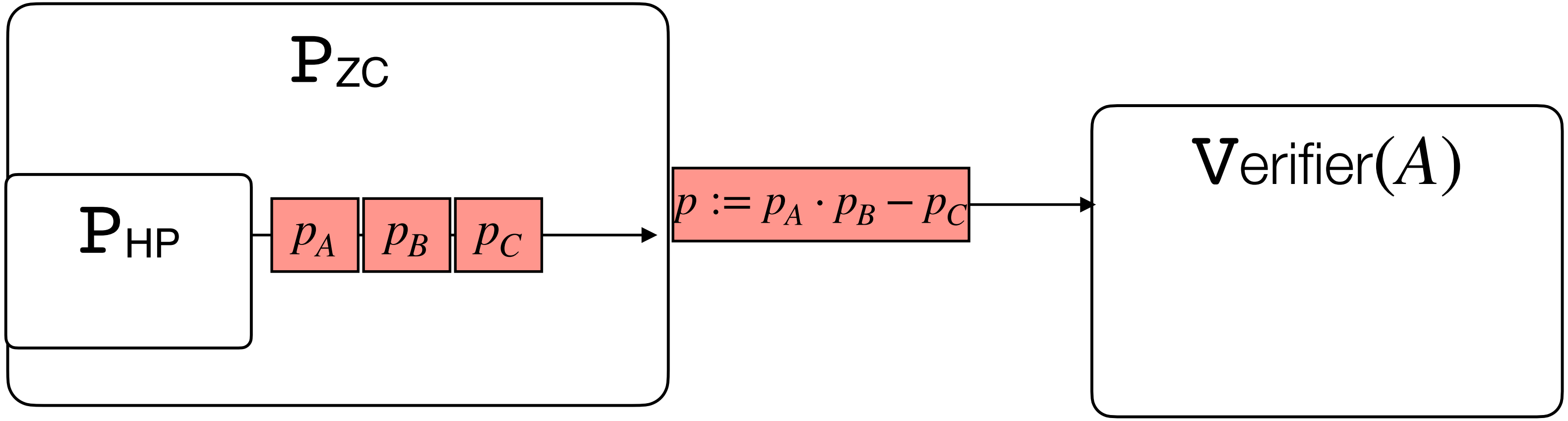
Run PIOP verifier for zerocheck for polynomial

$p_A \cdot p_B - p_C$.

Soundness

Strategy: Use adversary $\mathbf{P}_{\mathsf{HP}}$ against PIOP for HP

to get adversary $\mathbf{P}_{\mathsf{ZC}}$ against PIOP for ZeroCheck



If $\exists i$ such that $z_A[i] \cdot z_B[i] \neq z_C[i]$, then $p(h_i) \neq 0$, and so $p \neq 0$ on H, yet ZC verifier accepts, which breaks soundness of the PIOP for ZeroCheck.

What checks do we need?

Step 1: Correct Hadamard product
check that for each i, $z_A[i] \cdot z_B[i] = z_C[i]$

Step 2: Correct matrix multiplication
check that $Mz = z_M \quad \forall M \in \{A, B, C\}$

Starting point: *IP* for MV checks

Prover(M, z)

1. Compute $z_M := Mz$

z → z_M →

Verifier(M)

$r \xleftarrow{\$} \mathbb{F}$

$\vec{r} := (1, r, \ldots, r^{n-1})$

$\langle \vec{r}, z_M \rangle \stackrel{?}{=} \langle \vec{r}^\top M, z \rangle$

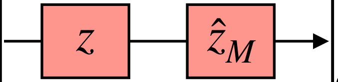
$$[\ \vec{r}\ ]\begin{bmatrix} z_M \end{bmatrix} \stackrel{?}{=} [\ \vec{r}\ ]\begin{bmatrix} M \end{bmatrix}\begin{bmatrix} z \end{bmatrix}$$

- **Soundness**: If there exists i such that $z_M[i] \neq Mz[i]$, then $\langle \vec{r}, z_M \rangle = \langle \vec{r}^\top M, z \rangle$ wp at most $1/|\mathbb{F}|$

Next point: *PIOP* for MV checks

Prover(M, z)

1. Compute $z_M := Mz$
2. Interpolate z_M over H to get $\hat{z}_M$



Verifier(M)

1. $r \xleftarrow{\$} \mathbb{F}$
2. $\vec{r} := (1, r, \ldots, r^{n-1})$
3. Interpolate $(\vec{r}, \vec{r}^\top M)$ to get $(\hat{r}, \hat{r}_M)$

How to compute inner products $\langle \hat{r}, \hat{z}_M \rangle, \langle \hat{r}_M, \hat{z} \rangle$?

Sumcheck → Inner product check

For vectors, we have that $\langle \vec{a}, \vec{b} \rangle = \sum_{i=1}^{n} a_i b_i$

What if $(\vec{a}, \vec{b})$ are represented as their interpolations $(\hat{a}, \hat{b})$?

Ans: $\sum_{i=1}^{n} a_i b_i = \sum_{h \in H} \hat{a}(h) \cdot \hat{b}(h)$

Next point: *PIOP* for MV checks

Prover(M, z)

1. Compute $z_M := Mz$
2. Interpolate z_M over H to get $\hat{z}_M$
3. Interpolate $(\vec{r}, \vec{r}^\top M)$ to get $(\hat{r}, \hat{r}_M)$
4. Invoke sumcheck PIOP prover on

$$\hat{r}(X) \cdot \hat{z}_M(X) - \hat{r}_M(X) \cdot \hat{z}(X)$$

Prover → Verifier: z, $\hat{z}_M$

Verifier → Prover: r

Verifier(M)

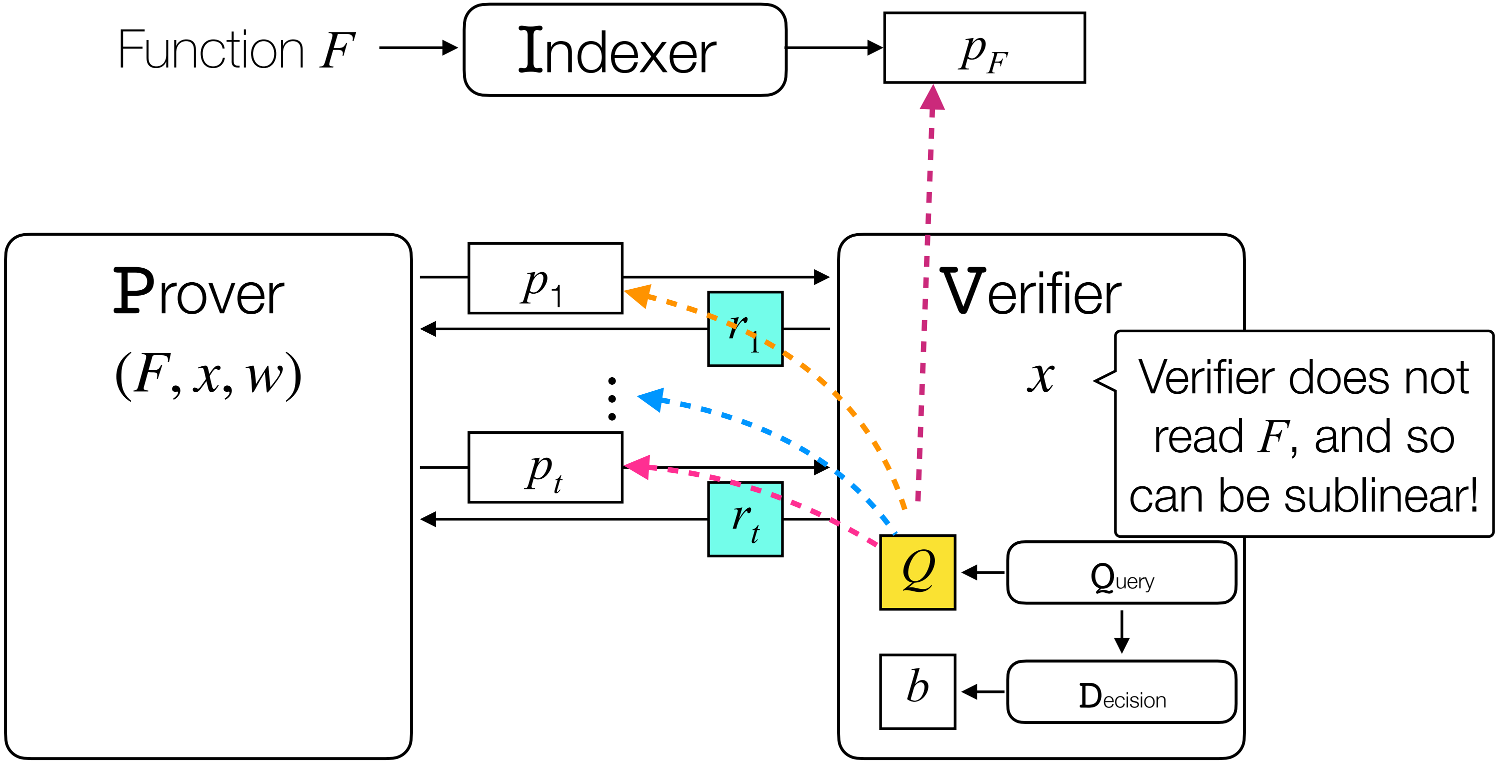
1. $r \xleftarrow{\$} \mathbb{F}$
2. $\vec{r} := (1, r, \ldots, r^{n-1})$
3. Interpolate $(\vec{r}, \vec{r}^\top M)$ to get $(\hat{r}, \hat{r}_M)$
4. Invoke sumcheck PIOP verifier on

$$\hat{r}(X) \cdot \hat{z}_M(X) - \hat{r}_M(X) \cdot \hat{z}(X)$$

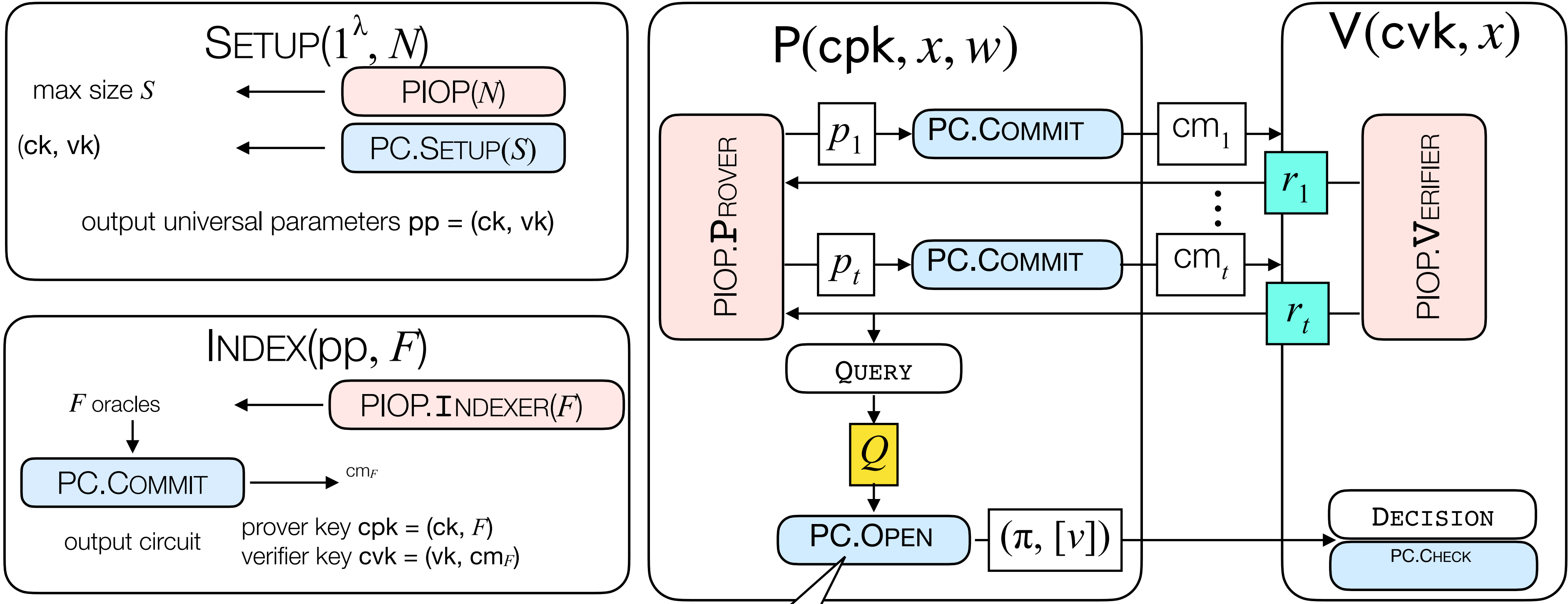
Sublinear verification for PIOP-based SNARKs

Holographic PIOPs [CHMMVW20, COS20]

Introduce a new algorithm to preprocess the circuit



Holographic PIOPs + PC Schemes → Preprocessing SNARKs



Prover answers queries to F oracles too

+ Fiat—Shamir to get non-interactivity

Verifier Complexity of Holographic PIOP-based SNARKs

$$T(\text{SNARK.V}) = T(\text{Check}) + T(\text{HIOP.}\mathbf{V})$$

Now sublinear!

Holography enables sublinear verification for arbitrary circuits computations!