Succinct Arguments

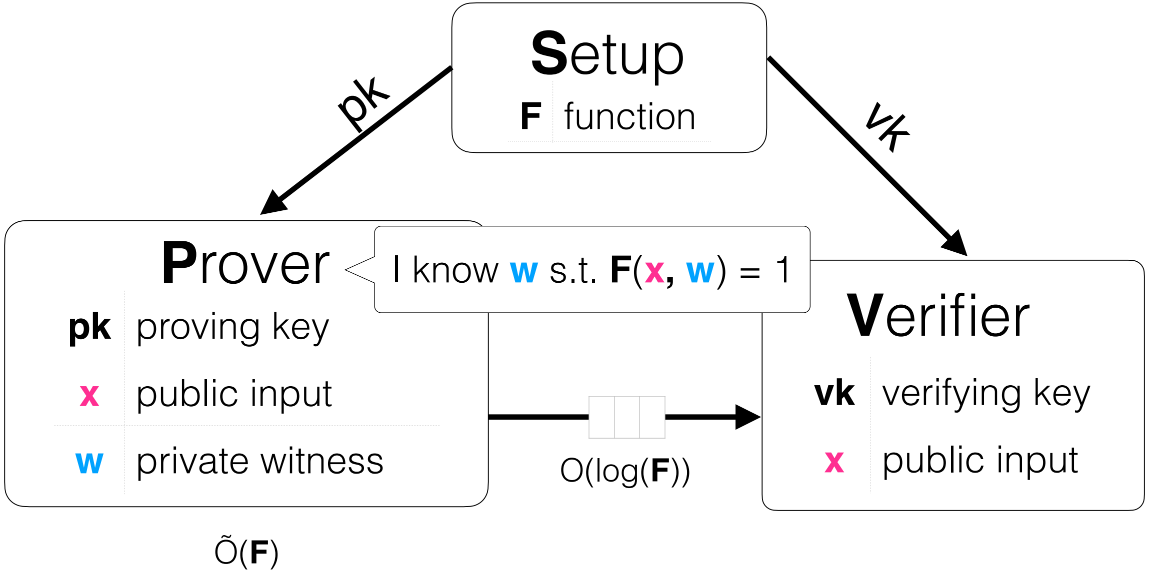
Lecture 03:
PIOP Toolkit and a PIOP for NP

Pratyush Mishra
UPenn
Fall 2025

Recap

Succinct Non-Interactive Arguments (SNARGs)

[Mic94, Groth10, GGPR13, Groth16…
…, GWC19, CHM**M**VW20, …]



Setup

F	function

pk

vk

Prover

pk	proving key
x	public input
w	private witness

I know **w** s.t. **F**(**x**, **w**) = 1

Verifier

vk	verifying key
x	public input

O(log(**F**))

Õ(**F**)

SNARKs

- **Completeness**: $\forall\ (F, x, w) \in \mathscr{R}$, $\Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) = 1 \ : \ \begin{array}{c} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathbf{P}(\mathsf{pk}, x, w) \end{array}\right] = 1.$

- **Knowledge Soundness**: $\forall$ efficient $\tilde{\mathbf{P}}$, $\exists$ extractor $\mathbf{E}$ s.t.

$$\Pr\left[\begin{array}{c} V(\mathsf{vk}, x, \pi) = 1 \\ \wedge \\ (F, x, w) \notin \mathscr{R} \end{array} \ : \ \begin{array}{c} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \tilde{\mathbf{P}}(\mathsf{pk}, x) \\ w \leftarrow \mathbf{E}_{\tilde{\mathbf{P}}}(\mathsf{pk}, x) \end{array}\right] \approx 0$$

- **Zero Knowledge**: $\exists$ **simulator** Sim s.t. $\forall\ (F, x, w) \in \mathscr{R}$, and all $\tilde{\mathbf{V}}$,

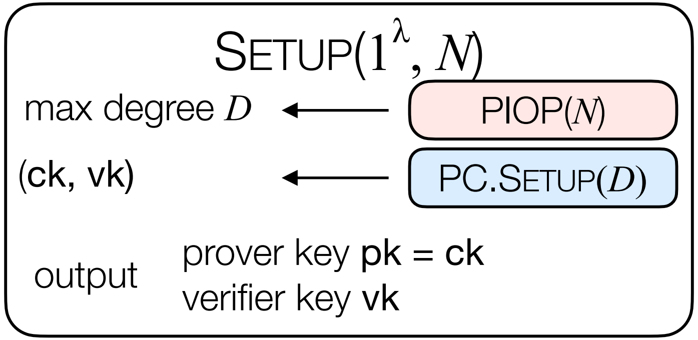
$$\Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) \ : \ \begin{array}{c} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathsf{Sim}(\mathsf{pk}, x) \end{array}\right] = \Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) \ : \ \begin{array}{c} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathbf{P}(\mathsf{pk}, x, w) \end{array}\right]$$

- **Succinctness**: $|\pi| = O(\log |F|)$ and $\mathsf{Time}(\mathbf{V}) = O(\log |F|, |x|)$

Constructing zkSNARKs

PIOP + PC = SNARK

PIOPs + PC Schemes → SNARK

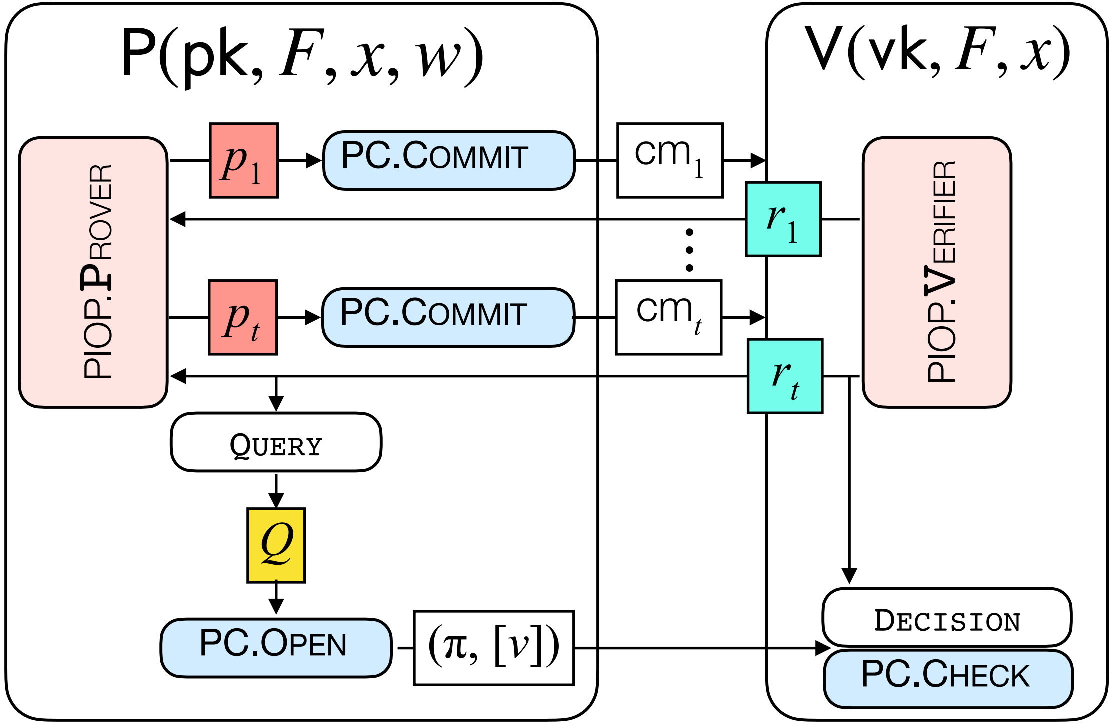


SETUP(1^λ, N)

max degree D ← PIOP(N)

(**ck**, **vk**) ← PC.SETUP(D)

output prover key **pk** = **ck**
verifier key **vk**



P(pk, F, x, w)

PIOP.PROVER

p_1 → PC.COMMIT → cm_1

r_1

p_t → PC.COMMIT → cm_t

r_t

QUERY

Q

PC.OPEN → (π, [v])

V(vk, F, x)

PIOP.VERIFIER

DECISION

PC.CHECK

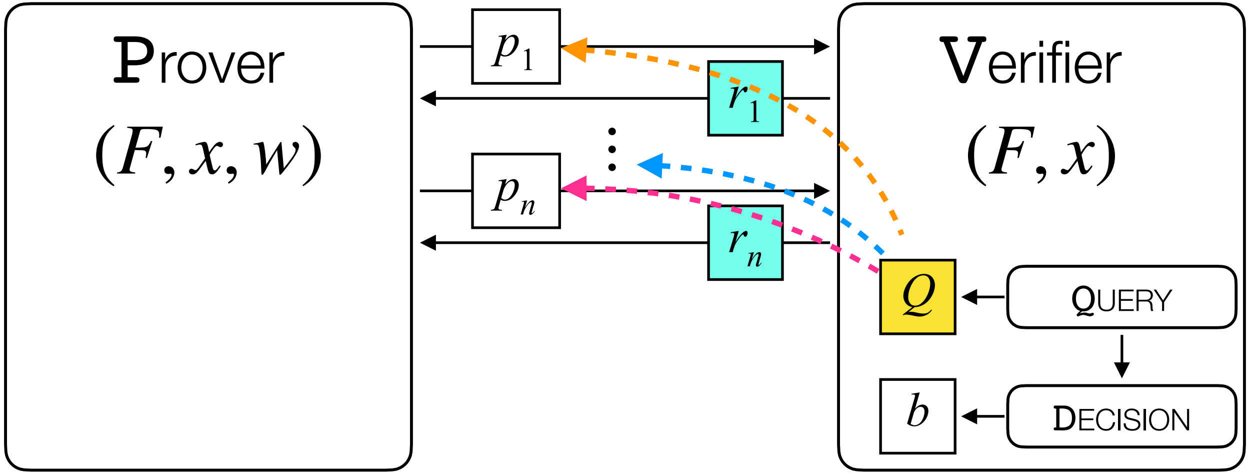
+ Fiat—Shamir to get non-interactivity

Verifier Complexity of PIOP-based SNARKs

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

Can make this sublinear (eg: KZG)

What about this?



PIOP Verifier has to **at least** read (F, x)

- When size of $F \ll$ size of computation (eg machine computations), TIME($\mathbf{V}$) **is sublinear.**
- When size of $F =$ size of computation (eg circuit computations), TIME($\mathbf{V}$) **is linear!**

Constructing PIOPs

Algebra background: Groups

Group: Set G equipped with an operation $*$

Identity: $\exists 1_G$ s.t. $\forall g \in G$ $1 * g = g * 1 = 1$

Inverse: for all $g \in G$, $\exists g^{-1}$ s.t. $g * g^{-1} = g^{-1} * g = 1$

Associative: $\forall a, b, c \in G$, $a * b * c = a * (b * c)$

Commutative: $\forall a, b \in G$, $a * b = b * a$

Lagrange's Thm

$\forall a \in G$,

$$a^{|G|} = 1$$

Examples:

→ Multiplicative subgroup of $\mathbb{Z}/p\mathbb{Z}$

→ Group of points on elliptic curve

Algebra background: Fields

Field : Set $\mathbb{F}$ equipped with 2 operations '+' and '×'

Group wrt '+' : $\mathbb{F}$ is an additive group w identity 0

Group wrt '×' : $\mathbb{F}^* = \mathbb{F} - \{0\}$ is a mult group w identity 1

Distributive : $\forall\, a, b, c :\quad a \times (b + c) = ab + ac$

Examples :

→ Real Numbers $\mathbb{R}$, Complex Numbers $\mathbb{C}$

→ Finite field : $\mathbb{Z}/p\mathbb{Z}$ ← Main structure we'll use

Algebra background: Polynomials

A univariate polynomial p over a field $\mathbb{F}$ in variable X

$$p(X) = \sum_{i=0}^{d} a_i X^i \; ; \qquad a_i \in \mathbb{F},\ d \in \mathbb{N}$$

degree

coefficients

We say $p \in \mathbb{F}^{\leq d}[X]$ → d-dimensional vector space over $\mathbb{F}$

A multivariate poly generalizes this to multiple variables

$$p(X_1, \dots, X_n) = \sum_{i=0}^{k} a_i X_1^{e_i^1} \cdots X_n^{e_i^n}$$

Individual deg of X_j = $\max_i(e_i^j)$

Total deg of p = $\max_i(\prod_{j=1}^{n} e_i^j)$

Algebra background: Poly Interpolation

Let $A = (a_1, \dots, a_n) \in \mathbb{F}^n$ be a list of elements

Let $H \subseteq \mathbb{F}$ be a subset of size $n = \{h_1, \dots, h_n\}$

Then we can use Lagrange Interpolation to find poly p s.t.

$$p(h_i) = a_i \quad \forall\, h_i \in H$$

How to construct p?

$$p(X) = \sum_{i=1}^{n} a_i \, \delta_H^{h_i}(X) \longleftarrow \begin{cases} 1 & \text{at } h_i \\ 0 & \text{elsewhere} \end{cases}$$

$$\uparrow$$

$$\prod_{h \in H - \{h_i\}} \frac{(X - h)}{(h_i - h)}$$

Algebra background: Poly Interpolation

$$p(X) = \sum_{i=1}^{n} a_i \, \delta_H^{h_i}(X) \leftarrow \prod_{h \in H - \{h_i\}} \frac{(X-h)}{(h_i-h)}$$

What is the time complexity of interpolation? Hint: each $\delta_H^{h_i}(X)$ is of degree $n-1$

Ans: $\underline{O(n^2)}$ [sum of n deg $n-1$ polys]

Can we do better?

A: Yes, when H is a $\underline{\text{multiplicative subgroup of size } 2^{\log_2 n}}$

$$H = \{1, \omega, \omega^2, \ldots, \omega^{n-1}\}$$

n-th root of unity

↪ can invoke IFFT to interpolate poly in $\boxed{O(n \log n)}$

↑ works like IFFT over $\mathbb{C}$

Background on univariate polynomials

Polynomial over $\mathbb{F}$:

$p(X) = a_0 + a_1X + \ldots + a_dX^d$ where $a_i \in \mathbb{F}$ and X takes values in $\mathbb{F}$.

Vanishing polynomial:

The vanishing polynomial for $H \subseteq \mathbb{F}$ is $v_H(X)$ such that $v_H(h) = 0 \;\; \forall\, h \in H$

Lagrange Polynomial:

The i-th Lagrange polynomial L_i is a polynomial that is 1 at h_i and 0 everywhere else in H. It is of the form $L_i(X) = c_i \cdot v_H(X)/(X - h_i)$.

Polynomial Interpolation:

Given $A = (a_0, \ldots, a_d)$, we can interpolate A over H to obtain $p(X)$ such that $p(h_i) = a_i$ where h_i is the i-th element of H. In particular, $p(X) = \sum a_iL_i(X)$.

Background on multilinear polynomials

Polynomial over $\mathbb{F}$***:***

$p(X_1, \ldots, X_n) = \sum_{i=0}^{2^n-1} a_i T_i$ where $a_i \in \mathbb{F}$ and T_i is a product of some of the X_j's.

Boolean Hypercube: The set $\{0,1\}^n$.

Lagrange polynomial:

The i-th Lagrange polynomial for the hypercube is the polynomial of the form $\mathrm{eq}_i(X_1, \ldots, X_n) = \prod_{j=1}^{n} \left((1 - i_j)(1 - X_j) + i_j X_j \right)$. This is 1 when X_i's form the Boolean decomposition of i.

Formalism of Relations

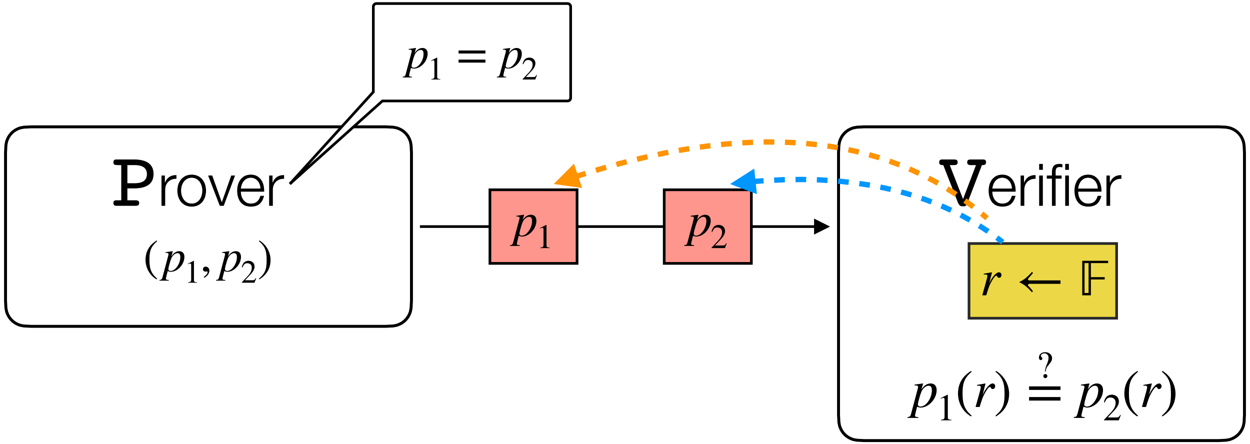
An NP Relation will be defined as a tuple:

$$(\mathbf{i}, (\mathbf{x}, \mathbf{y}), \mathbf{w})$$

- $\mathbf{i}$ is the NP *index*. Eg: circuit description
- $(\mathbf{x}, \mathbf{y})$ form the NP *instance*
    - $\mathbf{x}$ is the *explicit* instance
    - $\mathbf{y}$ is the *implicit* instance that is provided as an oracle
- $\mathbf{w}$ is the NP *witness*

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.

Schwartz-Zippel-DeMillo-Lipton Lemma

Lemma: Let $p(X_1, \ldots, X_n) \in \mathbb{F}[X_1, \ldots, X_n]$ be an ℓ-variate degree d polynomial. Then $\Pr_{r_1,\ldots,r_n \leftarrow \mathbb{F}}[p(r_1, \ldots, r_n) = 0] = \frac{d}{|\mathbb{F}|}$

Proof: Via induction on number of variables n

Base case: $n = 1$ follows from prior discussion

Hypothesis: Assume holds for $n - 1$ variables.

$\deg(p_i) \leq d - i$

Then, we can write $p(X_1, \ldots, X_n) := \sum_{i=1}^{d} X_1^i p_i(X_2, \ldots, X_n)$

For random $r_2, \ldots, r_n$, $\Pr[p_i(r_2, \ldots, r_n) = 0] = (d - i)/|\mathbb{F}|$.

Also, $\Pr[p(r_1, r_2, \ldots, r_n) = 0 \ \mid \ p_i(r_2, \ldots, r_n) \neq 0] = i/|\mathbb{F}|$

Then, $\Pr[E_n] = \Pr[E_n \cap E_{n-1}] + \Pr[E_n \cap \overline{E_{n-1}}]$

$$\leq \Pr[E_{n-1}] + i/|\mathbb{F}|$$

$$= \frac{d}{|\mathbb{F}|}$$

Sumcheck [LFKN90]

Protocols for the relation $\mathscr{R}_{\text{sum}}$ with

- $\mathbf{i} = \perp$
- $\mathbf{x} = (\sigma, S)$ where σ is a claimed sum and S is a subset of the field
- $\mathbf{y} = p$ is a polynomial
- $\mathbf{w} = \perp$

Multivariate Sumcheck [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:

$$\sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} p(b_1, \ldots, b_n) = \sigma_1 .$$

Sumcheck Protocol [LFKN90]

- **Start**: The protocol must check:

$$\sigma = \sigma_1 = \sum_{b_1 \in \{0,1\}} \dots \sum_{b_n \in \{0,1\}} p(b_1, \dots, b_n)$$

- **Round 1**:
  - P sends **univariate** polynomial $s_1(X_1)$ claimed to equal:

$$H(X_1) := \sum_{b_2 \in \{0,1\}} \dots \sum_{b_n \in \{0,1\}} p(X_1, b_2, \dots, b_n)$$

  - V checks that $\sigma_1 = s_1(0) + s_1(1)$.

Completeness: If $\sigma_1 = \sum_{b_1 \in \{0,1\}} \dots \sum_{b_n \in \{0,1\}} p(b_1, \dots, b_n)$ then $\sigma_1 = s_1(0) + s_1(1)$

Soundness: How can V check that $s_1 = H_1$?

Standard idea: Check that $s_1(r_1)$ = $H_1(r_1)$ for random point r_1.

V can compute $s_1(r)$ directly from P's first message, but not $H_1(r_1)$. What to do?

Idea: Recursion!

$$H(r_1) := \sum_{b_2 \in \{0,1\}} \dots \sum_{b_n \in \{0,1\}} p(r_1, b_2, \dots, b_n)$$

This is another sumcheck claim, over $n - 1$ variables!

Recursive Sumcheck [LFKN90]

- **Start**: The protocol must check:

$$\sigma = \sigma_1 = \sum_{b_1 \in \{0,1\}} \dots \sum_{b_n \in \{0,1\}} p(b_1, \dots, b_n)$$

- **Round 1**:
  - P sends **univariate** polynomial $s_1(X_1)$ claimed to equal:

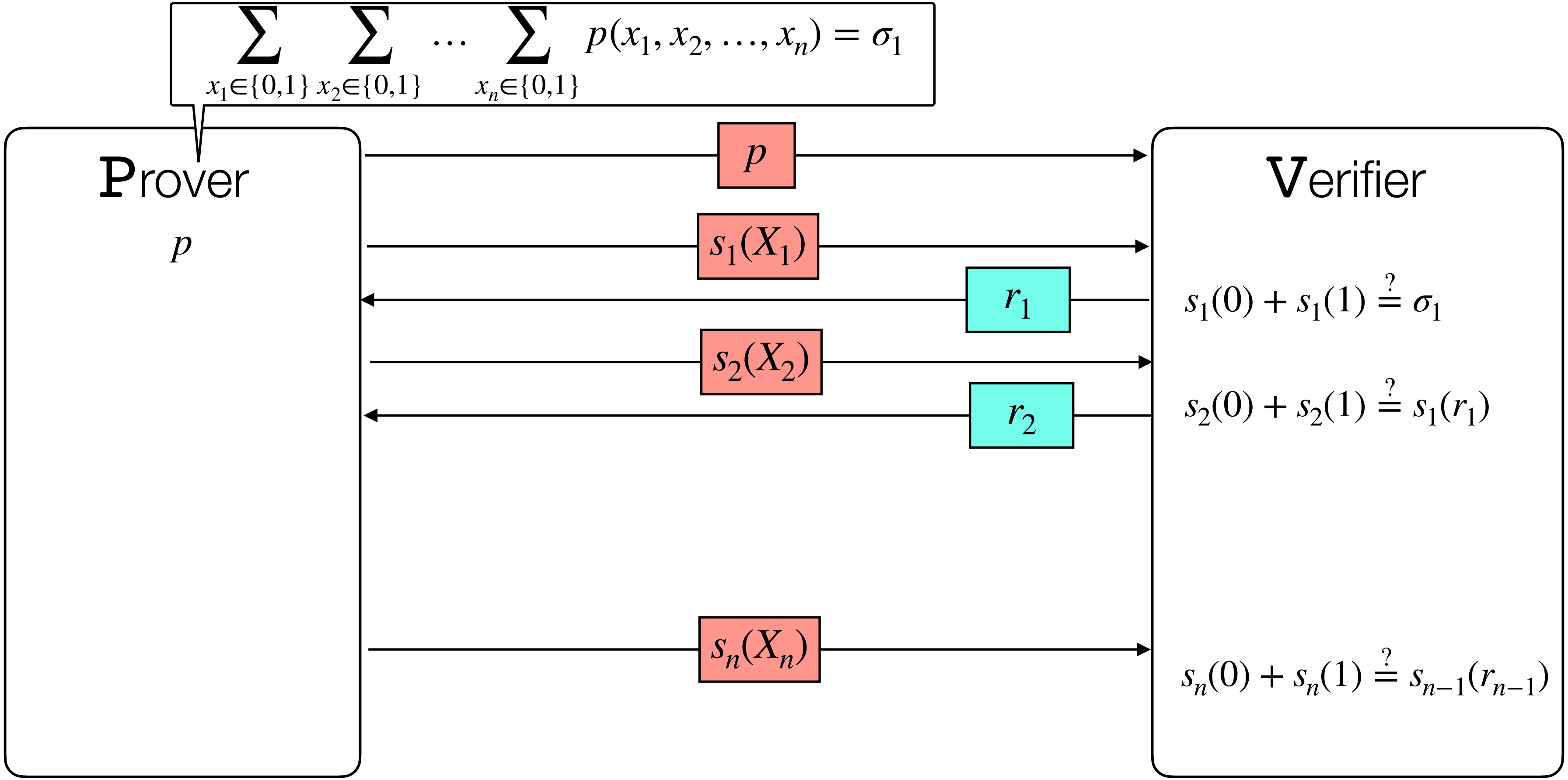
$$H(X_1) := \sum_{b_2 \in \{0,1\}} \dots \sum_{b_n \in \{0,1\}} p(X_1, b_2, \dots, b_n)$$

  - V checks that $\sigma_1 = s_1(0) + s_1(1)$ and sends $r_1 \overset{\$}{\leftarrow} \mathbb{F}$.
- **Round 2**:
  - P sends **univariate** polynomial $s_2(X_2)$ claimed to equal:

$$H_2(X_2) := \sum_{b_3 \in \{0,1\}} \dots \sum_{b_n \in \{0,1\}} p(r_1, X_2, b_3, \dots, b_n)$$

  - V checks that $s_1(r_1) = s_2(0) + s_2(1)$ and sends $r_2 \overset{\$}{\leftarrow} \mathbb{F}$.

Sumcheck protocol



$$\sum_{x_1\in\{0,1\}}\sum_{x_2\in\{0,1\}}\cdots\sum_{x_n\in\{0,1\}} p(x_1, x_2, \ldots, x_n) = \sigma_1$$

Prover

p

p

$s_1(X_1)$

r_1

$s_2(X_2)$

r_2

$s_n(X_n)$

Verifier

$s_1(0) + s_1(1) \overset{?}{=} \sigma_1$

$s_2(0) + s_2(1) \overset{?}{=} s_1(r_1)$

$s_n(0) + s_n(1) \overset{?}{=} s_{n-1}(r_{n-1})$

Completeness

We already saw that if Prover is honest, then the checks in a given round will pass.

So if P is honest in all rounds, all checks will pass

Soundness

Claim:

If P does not send the prescribed messages,
then V rejects with probability at least $1 - \frac{n \cdot d}{|\mathbb{F}|}$

(d is the maximum degree of p)

Soundness

Proof is by induction on the number of variables ℓ.

Base case: $n = 1$ In this case, P sends a single message $s_1(X_1)$ claimed to equal $p(X_1)$; V picks r_1 at random, checks that $s_1(r_1) = p(r_1)$

$$\text{If } s_1 \neq p, \text{ then } \Pr_{r_1 \in \mathbb{F}}[\, s_1(r_1) = p(r_1) \,] \le \frac{d}{|\mathbb{F}|}.$$

Soundness

Inductive case: $\ell > 1$.

- Recall: P's first message $s_1\left(X_1\right)$ is claimed to equal
$$H_1(X_1) := \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} p(X_1, b_2, \ldots, b_n)$$
- Then V picks a random r_1 and sends r_1 to P. They (recursively) invoke sumcheck to confirm that $s_1\left(r_1\right) = H_1\left(r_1\right)$.
- If $s_1 \neq H_1$, then $\Pr_{r_1 \in \mathbb{F}}[s_1\left(r_1\right) = H_1(r_1)] \leq \dfrac{d}{|\mathbb{F}|}$.
- If $s_1\left(r_1\right) \neq H_1\left(r_1\right)$, P must prove a *false* claim in the recursive call.
  - Claim is about $g\left(r_1, X_2, \ldots, X_\ell\right)$, which is $n-1$ variate.
  - By induction, P convinces V in the recursive call with prob at most $\dfrac{d(n-1)}{|\mathbb{F}|}$.

Soundness analysis: wrap-up

Summary: if $s_1 \neq H_1$, V accepts with probability at most:

$$\Pr_{r_1 \in \mathbb{F}}[s_1(r_1) = H(r_1)]$$

$$+$$

$$\Pr_{r_2,\ldots,r_n \in \mathbb{F}}\left[\text{V accepts} \mid s_1(r_1) \neq H(r_1)\right].$$

$$\leq \frac{d}{|\mathbb{F}|} + \frac{d(n-1)}{|\mathbb{F}|} \leq \frac{dn}{|\mathbb{F}|}$$

Costs of the sumcheck protocol

- Total communication is $O(dn)$ field elements.
  - P sends n univariate polynomials of degree at most d .
  - V sends $n - 1$ messages, each consisting of one field element.

- V's runtime is: $O(dn + [\text{time to evaluate } p \text{ at random point}])$

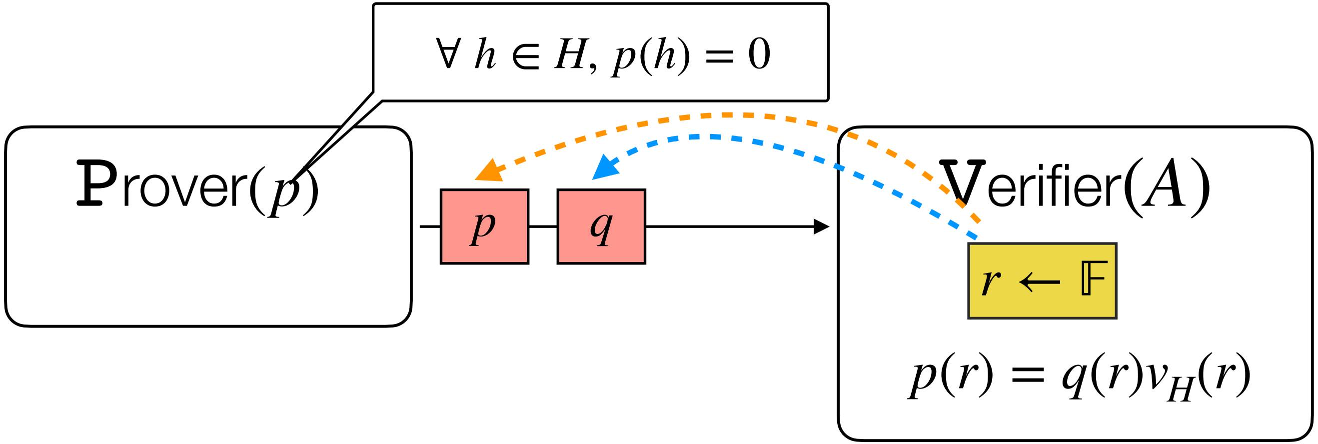
- P's runtime is at most: $O(d2^n + [\text{time to evaluate } p \text{ at random point}])$

Univariate Sumcheck [BCRSVW19]

- Input: V given oracle access to a univariate polynomial p over field $\mathbb{F}$ and claimed sum σ
- Goal: check the claim:

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

Univariate ZeroCheck



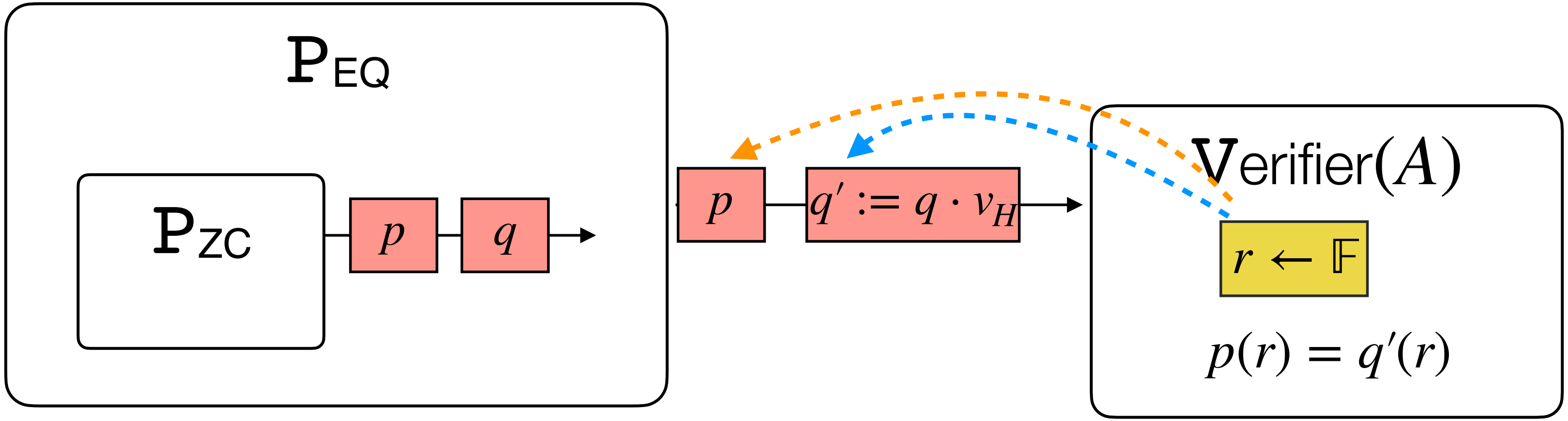
Lemma: $\forall h \in H,\ \ p(h) = 0$ if and only if $\exists q$ such that $p = q \cdot v_H$.

- **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.

Soundness

Strategy: Use adversary $\mathbf{P}_{\text{ZC}}$ against PIOP for ZeroCheck

to get adversary $\mathbf{P}_{\text{EQ}}$ against PIOP for Equality



- **Soundness**: If $p \neq q \,.\, v_H$, but $p(r) = q(r) \cdot v_H(r)$, then $\mathbf{P}_{\text{EQ}}$ breaks soundness of the PIOP for Equality. But this happens with negligible probability, so $\mathbf{P}_{\text{ZC}}$ is successful with negl. Probability.

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

Proof:

Special case where H is multiplicative subgroup consisting of roots of unity, and $\deg(p) = |H| - 1$. Then:

$$\sum_h p(h) = p(\omega^0) + p(\omega^1) + \cdots + p(\omega^{|H|-1})$$

$$= a_0 \cdot |H| + a_1 \cdot \sum \omega^i + a_2 \cdot \sum (\omega^2)^i + \cdots$$

Since sum of roots of unity is 0, so $\sum_h p(h) = \sigma = a_0 \cdot |H|$

Hence $\sigma / |H| = a_0$

Lemma: univariate sum check

Since $p(X) = a_0 + a_1 \cdot X + a_2X^2 + \cdots + a_{|H|-1}X^{|H|-1}$

And since $a_0 = \sigma / |H|$

Then we can write

$X \cdot g(X) = X \cdot (a_1 + a_2X + \cdots + a_{|H|-1}X^{|H|-2})$

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

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?

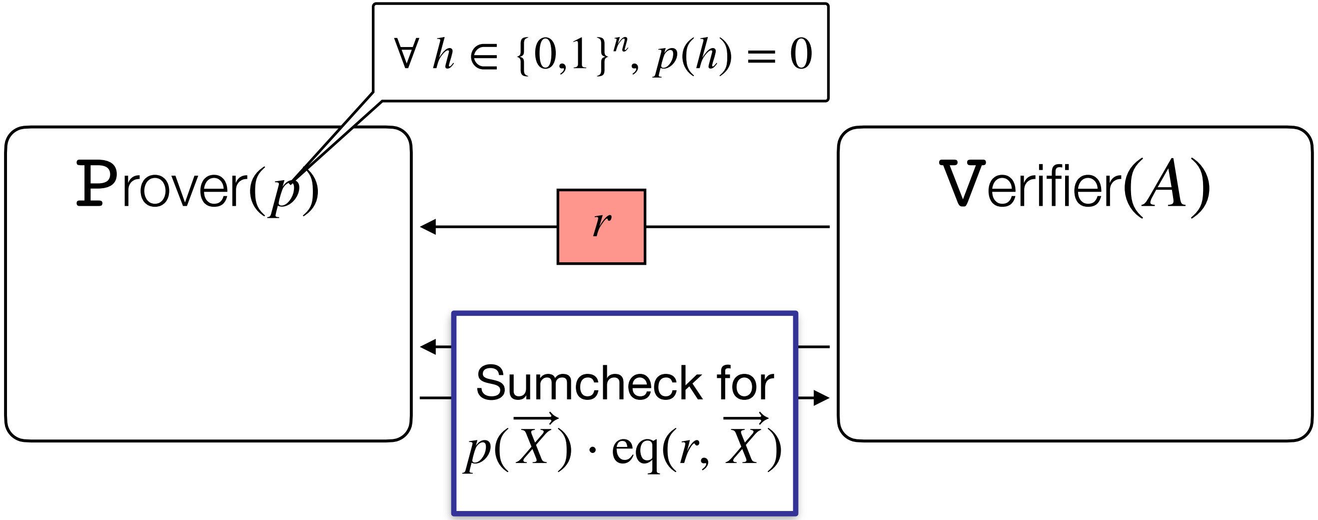
Zerocheck Protocol

- **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



$\forall\ h \in \{0,1\}^n,\ p(h) = 0$

Prover(p)

Verifier(A)

r

Sumcheck for $p(\overrightarrow{X}) \cdot \mathrm{eq}(r, \overrightarrow{X})$