Theory and Practice of Succinct Zero Knowledge Proofs

Lecture 09:
SNARKs from Linear PCPs

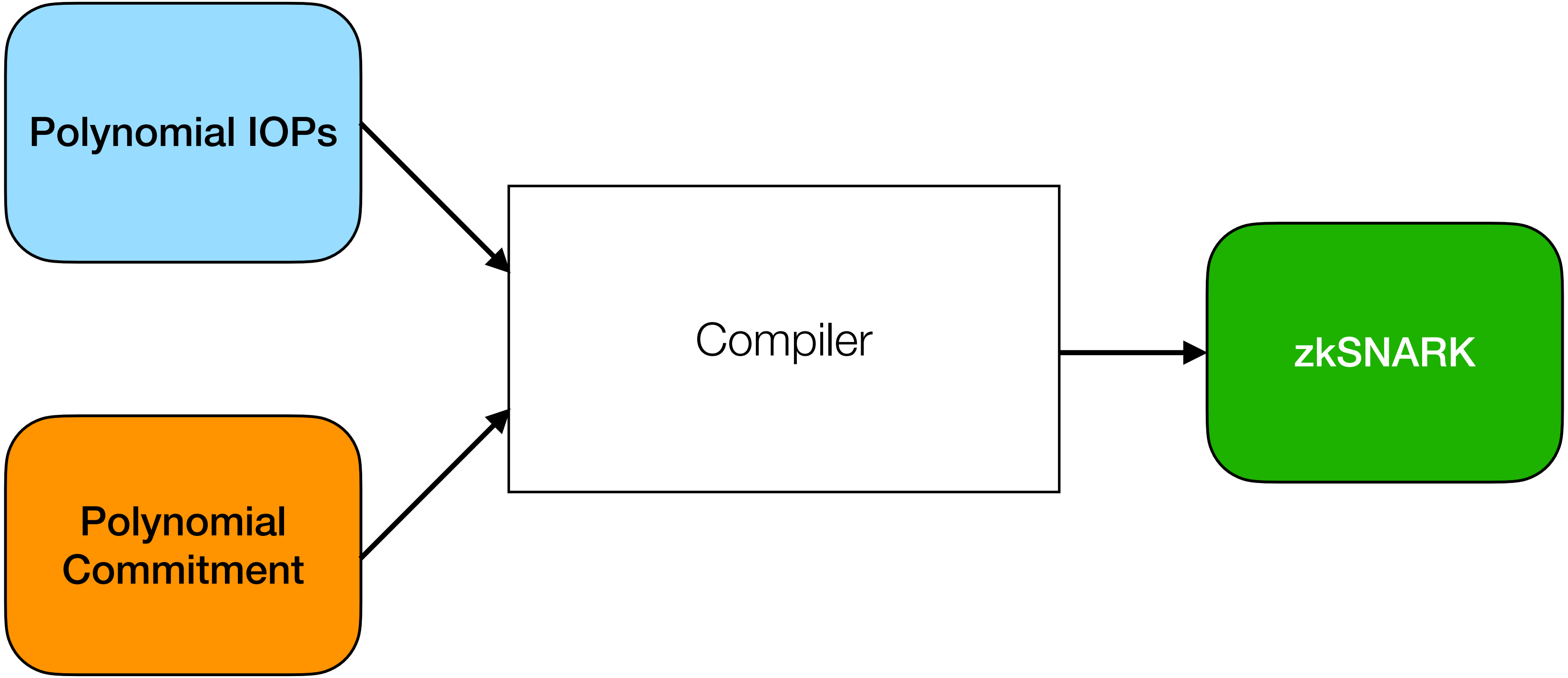
Pratyush Mishra
UPenn
Fall 2023

Announcements

- **Next assignment due Monday 10/09 midnight**
- **Next discussion-oriented class 10/10**
    - If you're presenting, reach out to me by this Friday 10/06!
- **Project:**
    - Project proposal **deadline is 10/10**!
    - Talk to me if you'd like to chat about project topics

Recap

PIOP + PC = SNARK



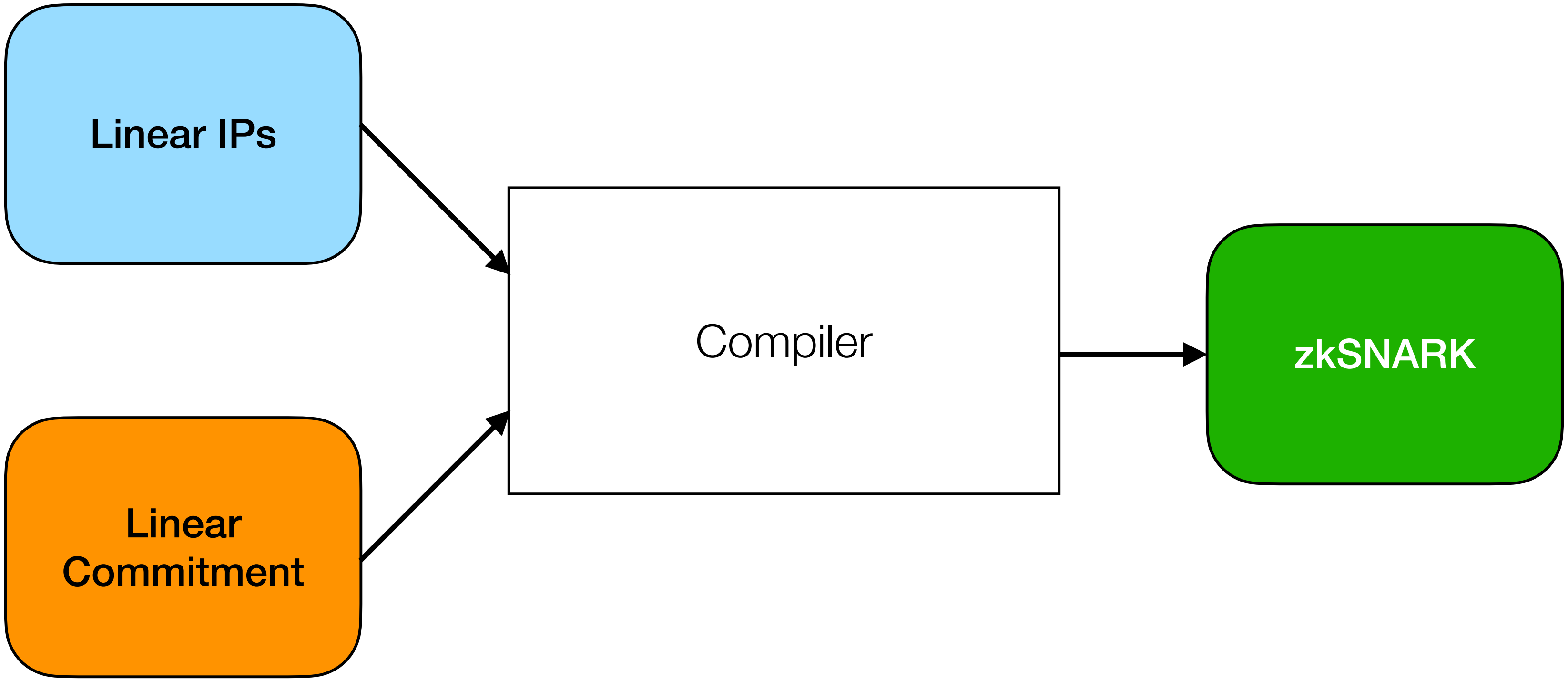
SNARKs So Far

PIOP	PC Scheme	Setup	P Time	V Time	Pf size
Marlin	KZG	Trusted	$O(n \log n)$	$O(\log n)$	~1 kB
Spartan	DL-based	Transparent	$O(n)$	$O(\text{sqrt}(n))$	10 -100kB

How small can verifier time and proof size be?

New Recipe:
LIPs
+
Linear Commitments

New Compiler



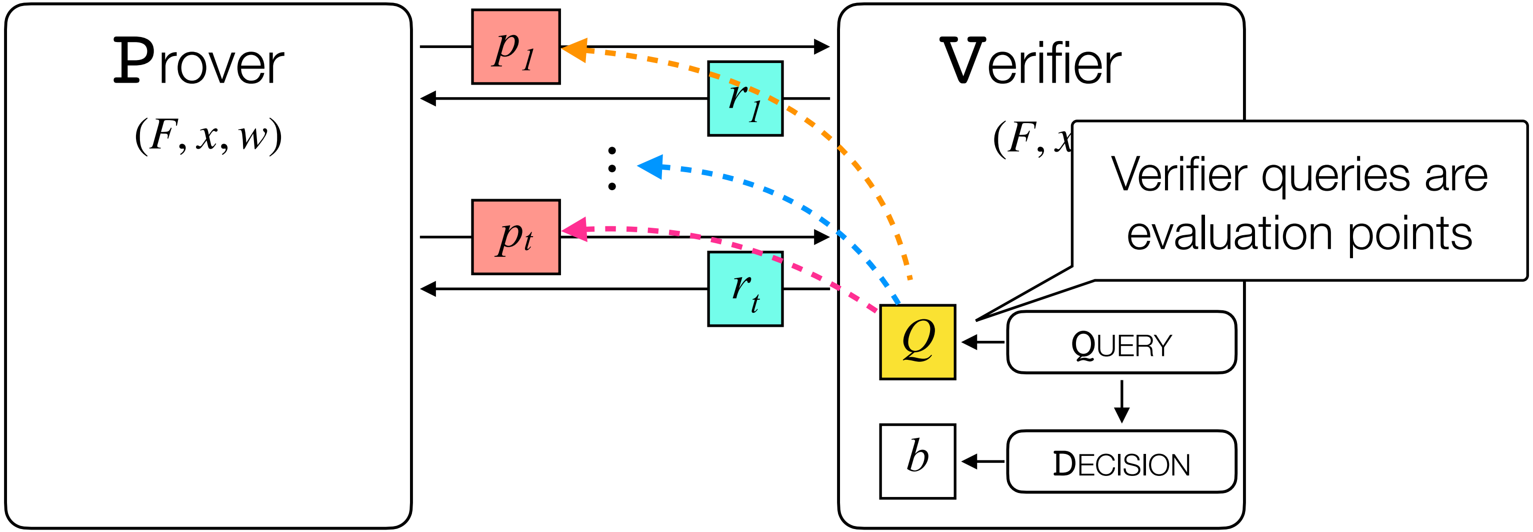
SNARK Comparison

PIOP	PC Scheme	Setup	P Time	V Time	Pf size
Marlin	KZG	Trusted	$O(n \log n)$	$O(\log n)$	~1 kB
Spartan	DL-based	Transparent	$O(n)$	$O(\text{sqrt}(n))$	10 -100kB

LIP	LC Scheme	Setup	P Time	V Time	Pf size
Groth16	GGM	Circuit-specific trusted	$O(n \log n)$	$\boldsymbol{O(1)}$	**< 200B**

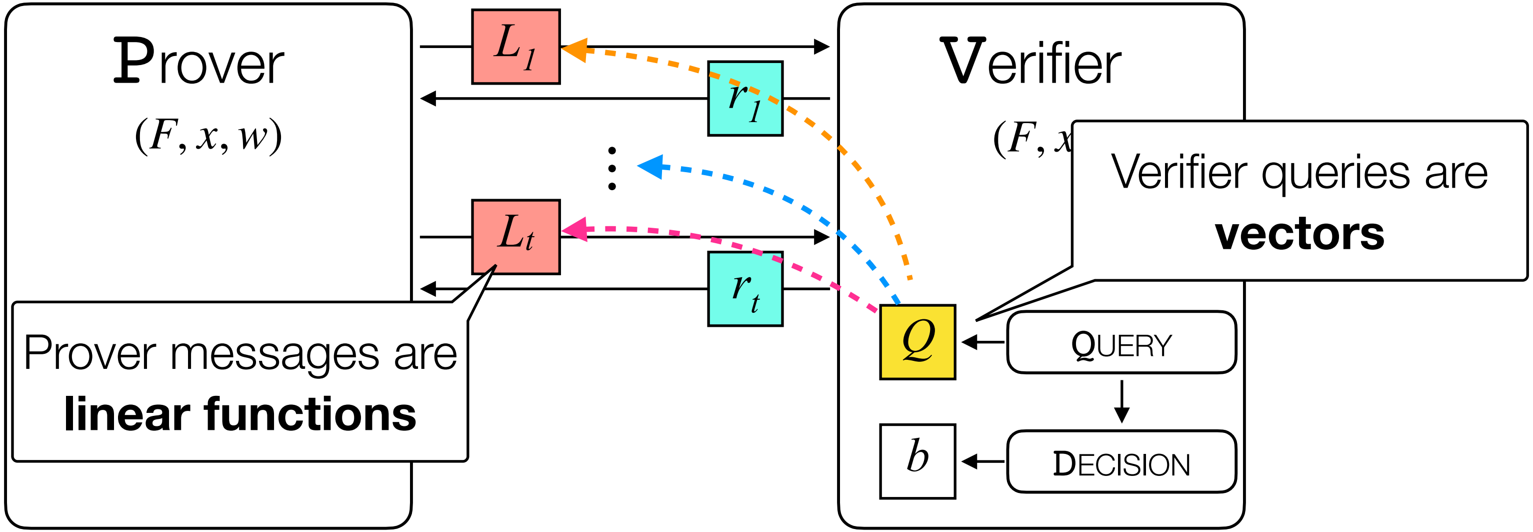
Definition:
Linear IP

Recall: PIOPs [GWC19, CHMMVW20, BFS20]



- **Completeness**: Whenever $(F, x, w) \in \mathscr{R}$, there is a strategy for $\mathbb{P}$ that outputs **only polynomials**, and which causes $\mathbb{V}$ to accept.
- **Knowledge Soundness**: Whenever $\mathbb{V}$ accepts against a $\mathbb{P}$ that outputs **only polynomials**, then $\mathbb{P}$ "knows" w such that $(F, x, w) \in \mathscr{R}$.
- **Bounded-query ZK**: Whenever $(F, x, w) \in \mathscr{R}$, a $\mathbb{V}$ that makes up to b queries to polys learns nothing about w.

New: Linear IOPs [GGPR13, BCIOP13, SBVBPW13]



- **Completeness**: Whenever $(F, x, w) \in \mathscr{R}$, there is a strategy for $\mathbb{P}$ that outputs **only linear functions**, and which causes $\mathbb{V}$ to accept.
- **Knowledge Soundness**: Whenever $\mathbb{V}$ accepts against a $\mathbb{P}$ that outputs **only linear functions**, then $\mathbb{P}$ "knows" w such that $(F, x, w) \in \mathscr{R}$.
- **Bounded-query ZK**: Whenever $(F, x, w) \in \mathscr{R}$, a $\mathbb{V}$ that makes up to b queries to polys learns nothing about w.

Construction:
Linear IP for R1CS

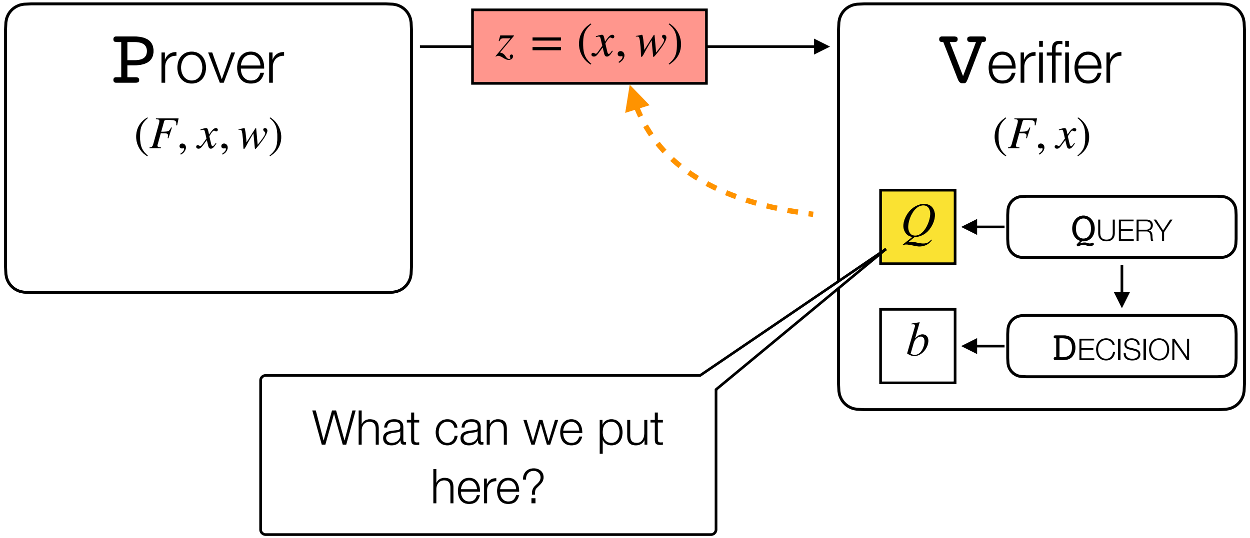
R1CS

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

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

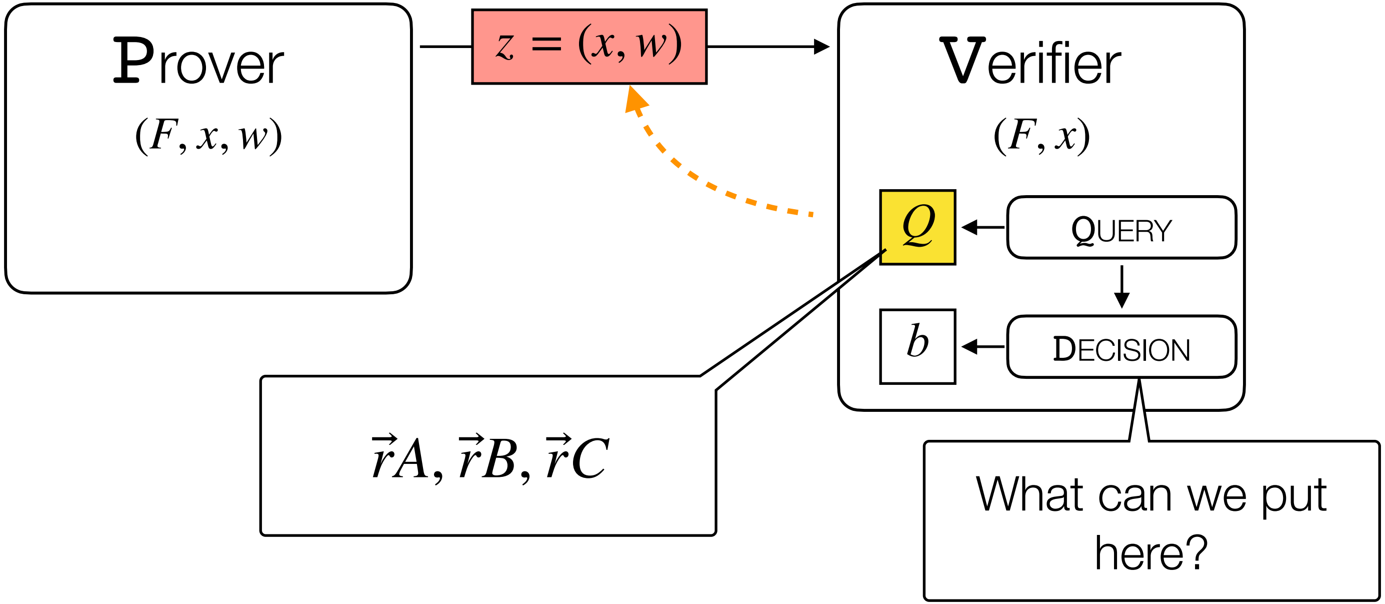
$$z := \begin{bmatrix} x \\ w \end{bmatrix} \qquad [A][z] \circ [B][z] = [C][z]$$

Attempt #1



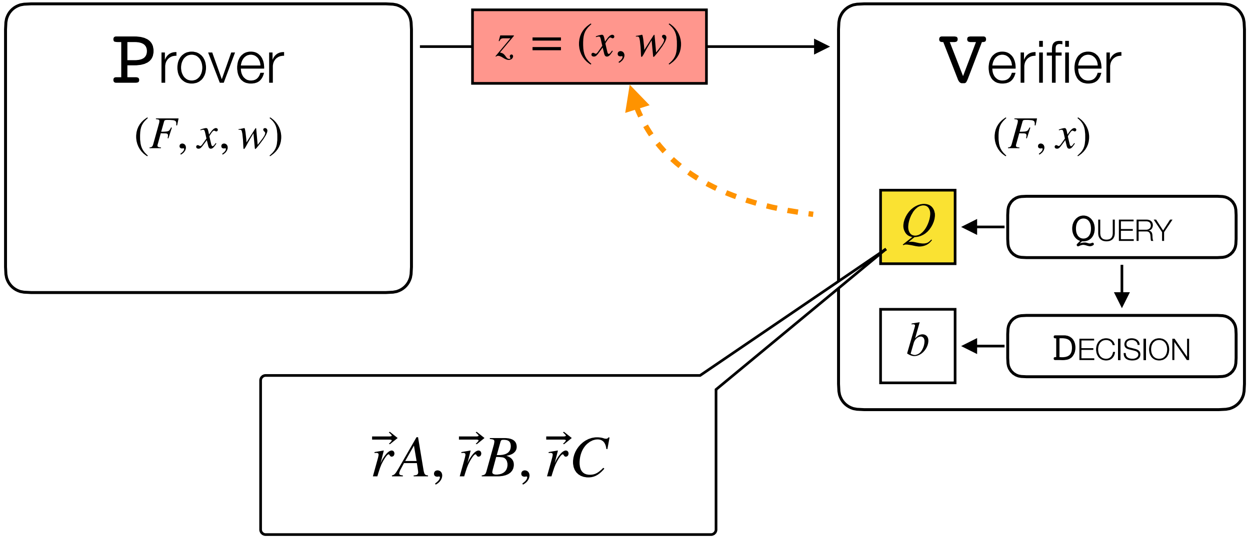
- Idea 1: Just a random vector: $\vec{r} := (1, r, r^2, \ldots, r^{n-1})$
  - $\langle z, \vec{r} \rangle$ doesn't seem that useful…

Attempt #2



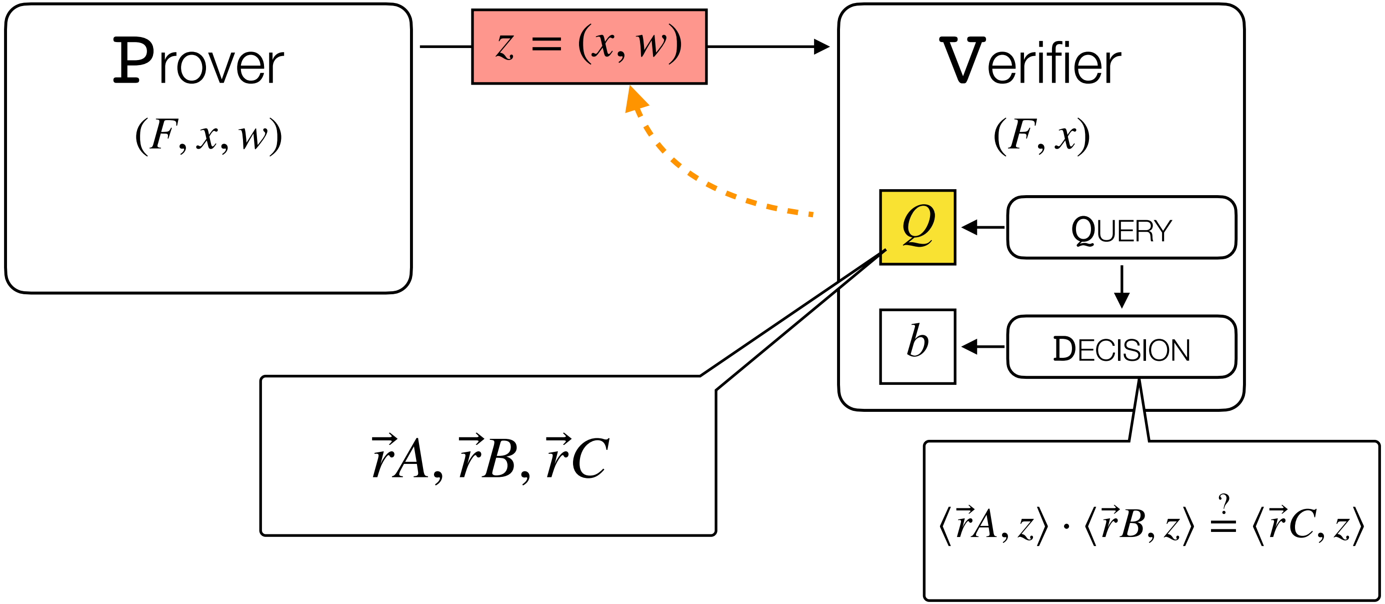
- Hint: Think of the lincheck PIOP!
- Idea 2: $\vec{r} \cdot M$ for each $M \in \{A, B, C\}$
  - What can we do with $\langle \vec{r}A, z \rangle, \langle \vec{r}B, z \rangle, \langle \vec{r}C, z \rangle$?

Attempt #2



- Hint: Think of the lincheck PIOP!
- Idea 2: $\vec{r} \cdot M$ for each $M \in \{A, B, C\}$
  - What can we do with $\langle \vec{r}A, z \rangle, \langle \vec{r}B, z \rangle, \langle \vec{r}C, z \rangle$?

Attempt #2



- Hint: Think of the lincheck PIOP!
- Idea 2: $\vec{r} \cdot M$ for each $M \in \{A, B, C\}$
    - How about checking the product?

Let's analyze this

$$\left[M\right] = \begin{bmatrix} \leftarrow m_1 \rightarrow \\ \leftarrow m_n \rightarrow \end{bmatrix}$$

$$\langle \vec{r}M, z\rangle = \langle \vec{r}, Mz\rangle$$
$$= \sum_i r^i \langle m_i, z\rangle$$

Then we have that

$$\langle \vec{r}A, z\rangle \cdot \langle \vec{r}B, z\rangle$$

$$= \left(\sum_i r^i \langle a_i, z\rangle\right) \cdot \left(\sum_j r^j \langle b_j, z\rangle\right)$$
$$= \sum_{i,j} r^{i+j} \cdot \langle a_i, z\rangle \cdot \langle b_j, z\rangle$$

Let's analyze this

$$= \sum_{i,j} r^{i+j} \cdot \langle a_i, z \rangle \cdot \langle b_j, z \rangle$$

$$= \sum_{i} r^{2i} \cdot \langle a_i, z \rangle \cdot \langle b_i, z \rangle + \sum_{i \neq j} r^{i+j} \cdot \langle a_i, z \rangle \cdot \langle b_j, z \rangle$$

$$= \langle \overrightarrow{r^2}, Cz \rangle + \text{junk}$$

Almost there!

We just have to get rid of … $O(n^2)$ junk terms 😞

Attempt #3: A Different Basis

We saw that, for each $M \in \{A, B, C\}$,

$$\langle \vec{r}M, z \rangle = \sum_i r^i \langle m_i, z \rangle$$

This looks like a polynomial!

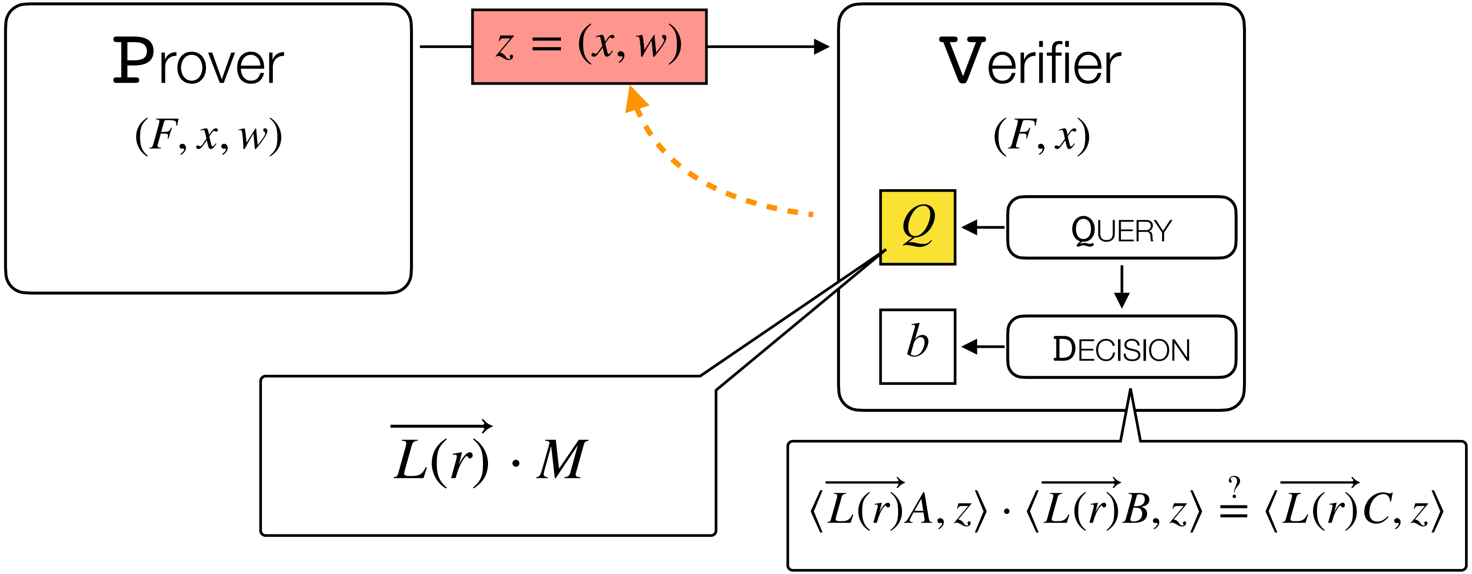
$$p_M(r) = \sum_i r^i \langle m_i, z \rangle$$

This is a polynomial in the *monomial basis.*

Using this basis didn't work.

What should we try next?

Attempt #3: Lagrange Basis!



- New idea: query for $\overrightarrow{L(r)} \cdot M := (L_1(r), L_2(r), \ldots, L_n(r)) \cdot M$
  - $L_i(X)$ is i-th Lagrange basis poly for n-sized domain H

Let's analyze this

$$\langle \overrightarrow{L(X)} M, z \rangle = \sum_i L_i(X) \langle m_i, z \rangle$$

Then we have that

$$\langle \overrightarrow{L(X)} A, z \rangle \cdot \langle \overrightarrow{L(X)} B, z \rangle$$

$$= \Big(\sum_i L_i(X) \langle a_i, z \rangle \Big) \cdot \Big(\sum_j L_j(X) \langle b_j, z \rangle \Big)$$

$$= \sum_{i,j} L_i(X) L_j(X) \cdot \langle a_i, z \rangle \cdot \langle b_j, z \rangle$$

Let's analyze this

$$= \sum_{i,j} L_i(X)L_j(X) \cdot \langle a_i, z \rangle \cdot \langle b_j, z \rangle$$

$$= \sum_{i} L_i(X)^2 \cdot \langle a_i, z \rangle \cdot \langle b_i, z \rangle + \sum_{i \neq j} L_i(X)L_j(X) \cdot \langle a_i, z \rangle \cdot \langle b_j, z \rangle$$

$$= \sum_{i} L_i(X)^2 \cdot \langle a_i, z \rangle \cdot \langle b_i, z \rangle + \text{junk}$$

Still stuck?!?!

What are we doing wrong?

Idea: Remember Hadamard PIOP

What does this remind you of?

$$\langle \overrightarrow{L(X)} M, z \rangle = \sum_i L_i(X) \langle m_i, z \rangle$$

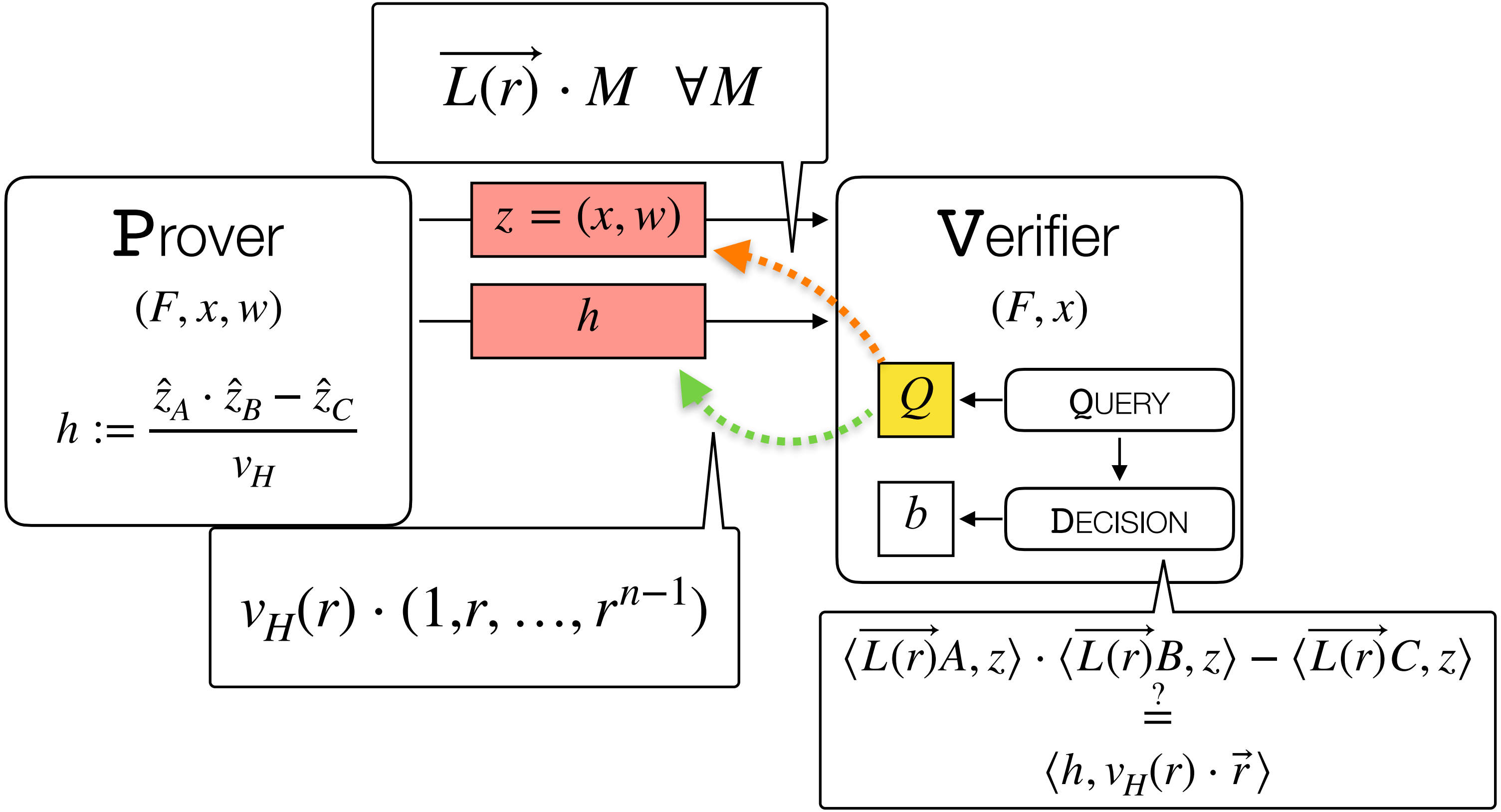
This is the interpolation of Mz over H!

So after queries we have $\hat{z}_A(r), \hat{z}_B(r), \hat{z}_C(r)$!

Q: What did we do with these in Hadamard PIOP?

A: Check $\hat{z}_A(r) \cdot \hat{z}_B(r) - \hat{z}_C(r) = h(r) \cdot v_H(r)$

Final Construction



Let's analyze this: Completeness

$$
\begin{aligned}
&= \Big(\sum_i L_i(X)\langle a_i, z\rangle\Big) \cdot \Big(\sum_j L_j(X)\langle b_j, z\rangle\Big) - \sum_j L_j(X) \cdot \langle c_j, z\rangle \\
&= \hat{z}_A(X) \cdot \hat{z}_B(X) - \hat{z}_C(X) \\
&= h(X) \cdot v_H(X)
\end{aligned}
$$

Let’s analyze this: Soundness

$\hat{z}_A(X) \cdot \hat{z}_B(X) - \hat{z}_C(X) \neq h(X) \cdot v_H(X)$,

then $\hat{z}_A(r) \cdot \hat{z}_B(r) - \hat{z}_C(r) = h(r) \cdot v_H(r)$

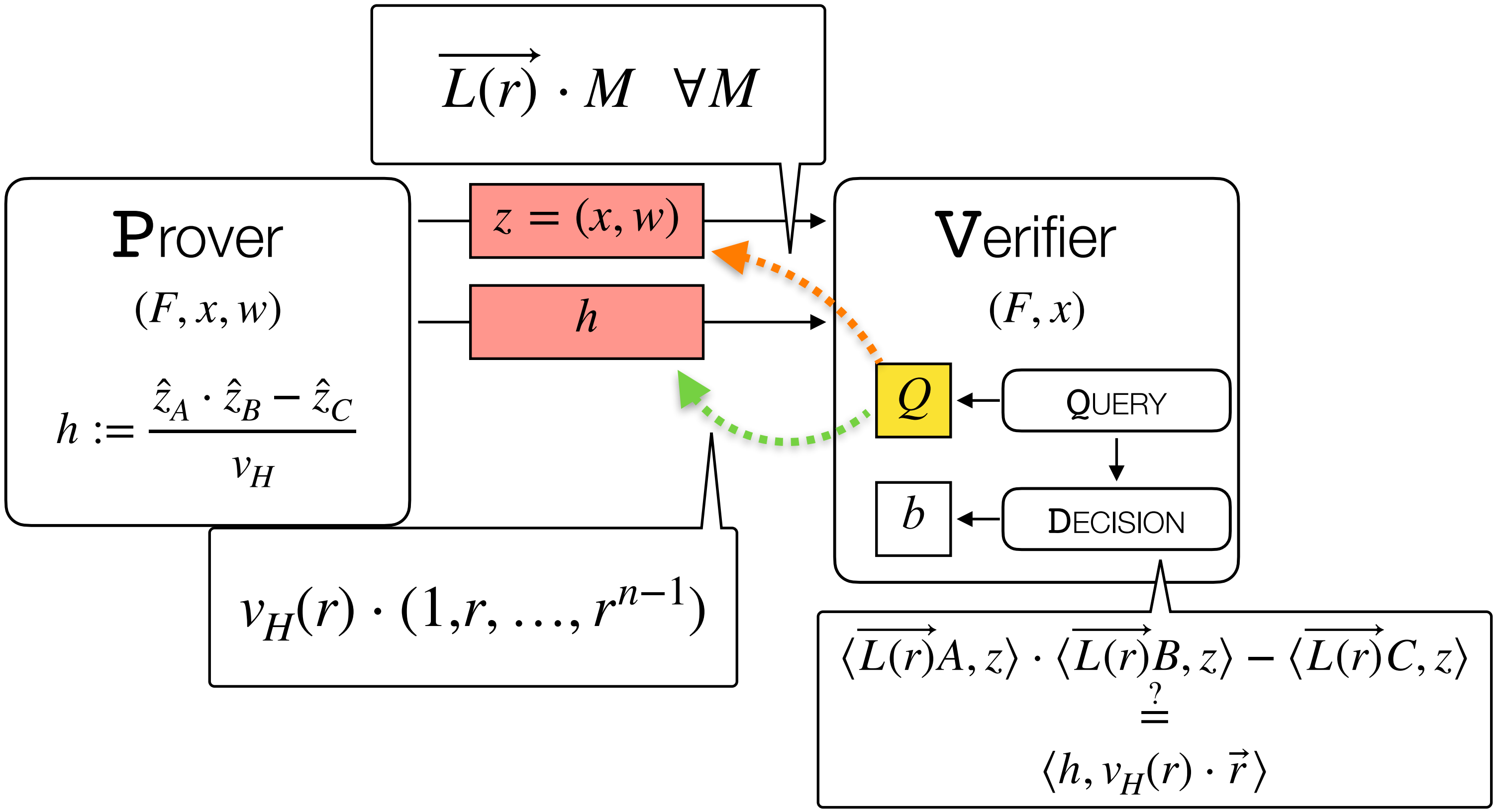
with negligible probability

Let's analyze this: Efficiency

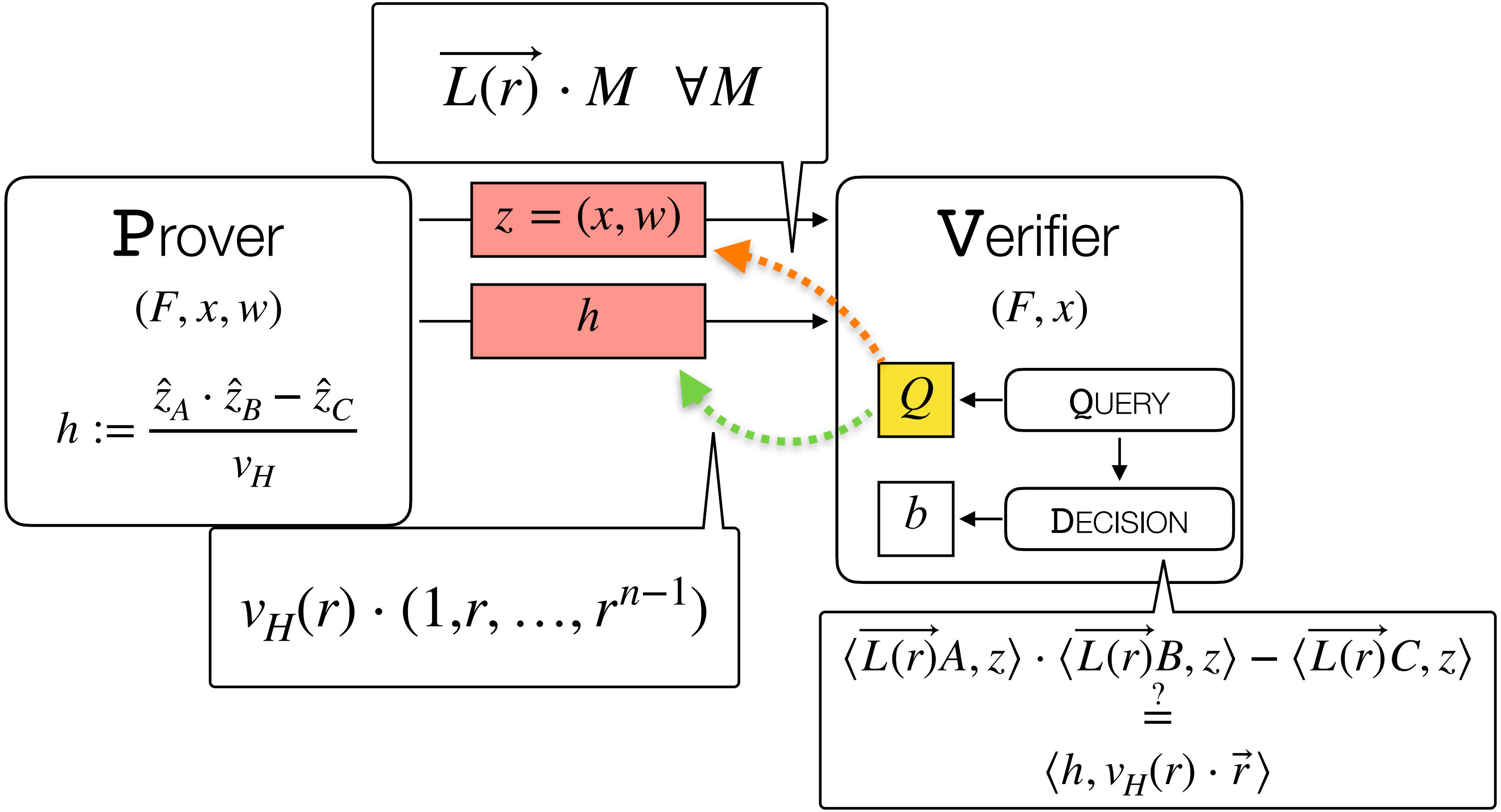
- Number of oracles: 2
- Number of queries: 4
- Prover work: $O(n \log n)$ (due to poly mul)
- Number of rounds: 1
- Verifier checks: ~1 multiplication

Compiling LIPs to SNARKs

Q: What is verifier computation?



A: $O(n)$!

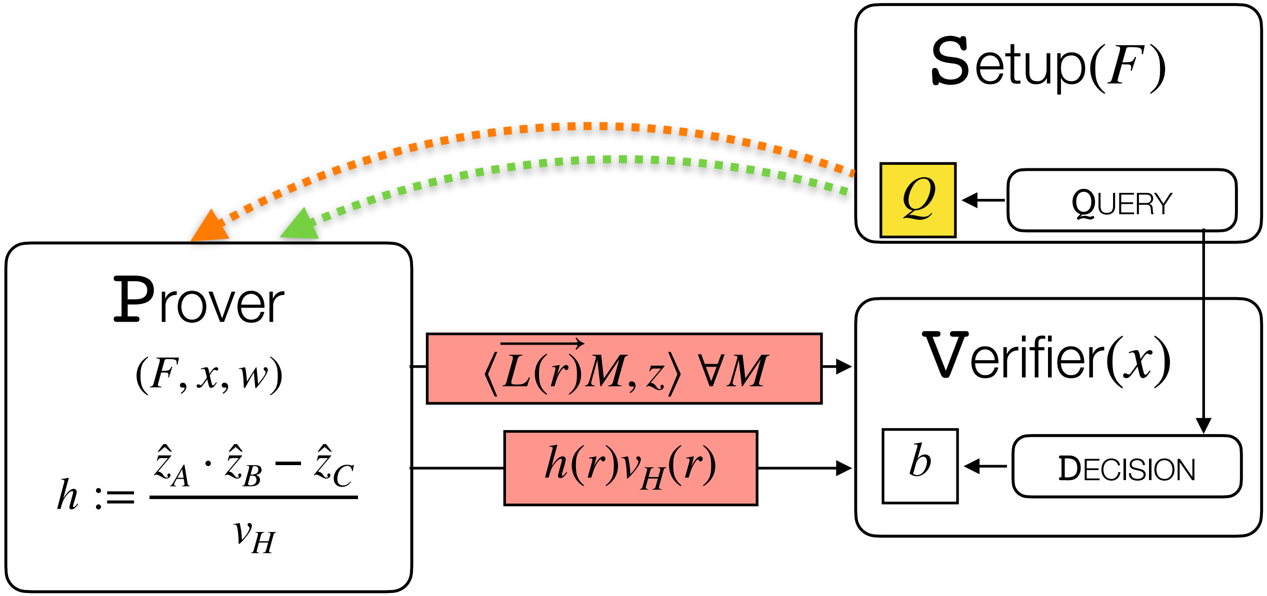


$\overrightarrow{L(r)} \cdot M \quad \forall M$
Prover
(F, x, w)
$h := \frac{\hat{z}_A \cdot \hat{z}_B - \hat{z}_C}{v_H}$
$z = (x, w)$
h
Verifier
(F, x)
Q
QUERY
b
DECISION
$v_H(r) \cdot (1, r, \ldots, r^{n-1})$
$\langle \overrightarrow{L(r)} A, z \rangle \cdot \langle \overrightarrow{L(r)} B, z \rangle - \langle \overrightarrow{L(r)} C, z \rangle \stackrel{?}{=} \langle h, v_H(r) \cdot \vec{r} \rangle$

Can we do better?

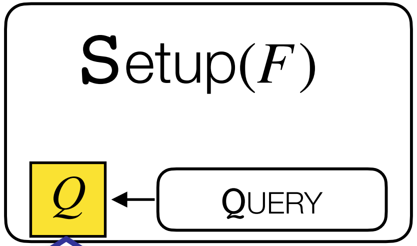
Yes, via preprocessing!

Insight: all queries are independent of prover message



- Problem: No soundness!

Idea: Encode in Exponent



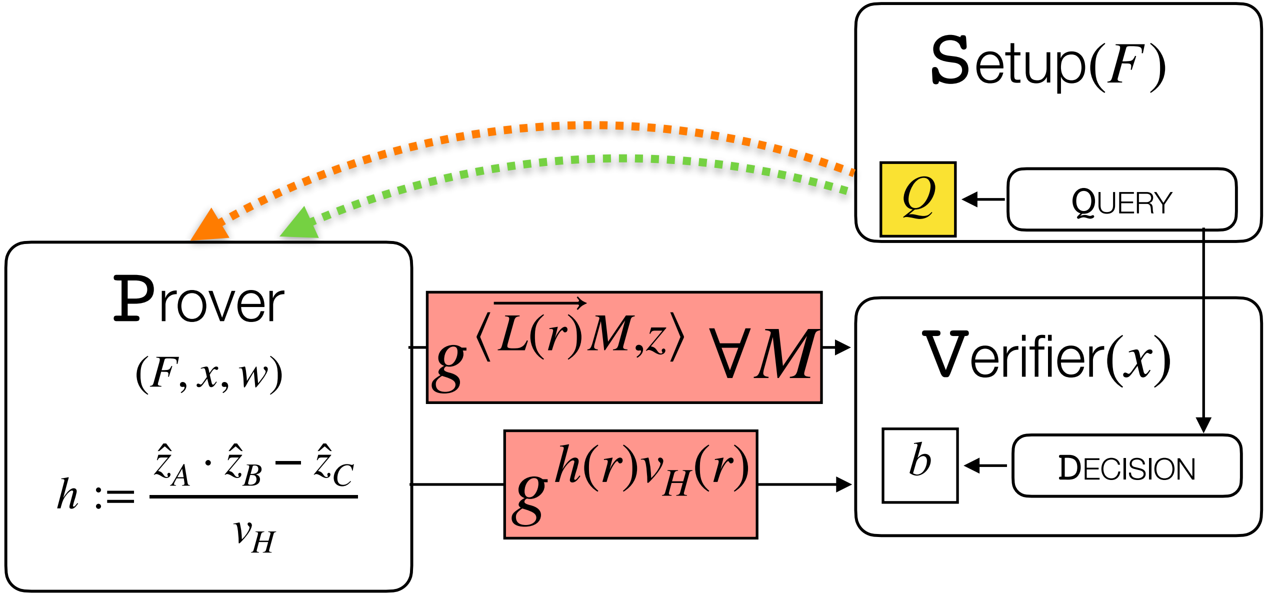
Let m_i be the i-th column of M.

Then $\hat{m}_i(X) := \langle \overrightarrow{L(X)}, m_i \rangle$ is its interpolation, and

$\overrightarrow{L(X)} \cdot M := (\hat{m}_1(X), \ldots, \hat{m}_n(X))$

- For each M, define $\mathsf{pk}_M := (g^{\hat{m}_1(r)}, \ldots, g^{\hat{m}_n(r)})$
- Define $\mathsf{pk}_h := (g^{v_H(r)}, g^{r \cdot v_H(r)}, \ldots, g^{r^{n-1} v_H(r)})$

Construction with encoded queries



Q: How to perform check in exponent?

Q: How to perform check in exponent?

We have

$$g_A := g^{\langle \overrightarrow{L(r)} A, z \rangle}$$

$$g_B := g^{\langle \overrightarrow{L(r)} B, z \rangle}$$

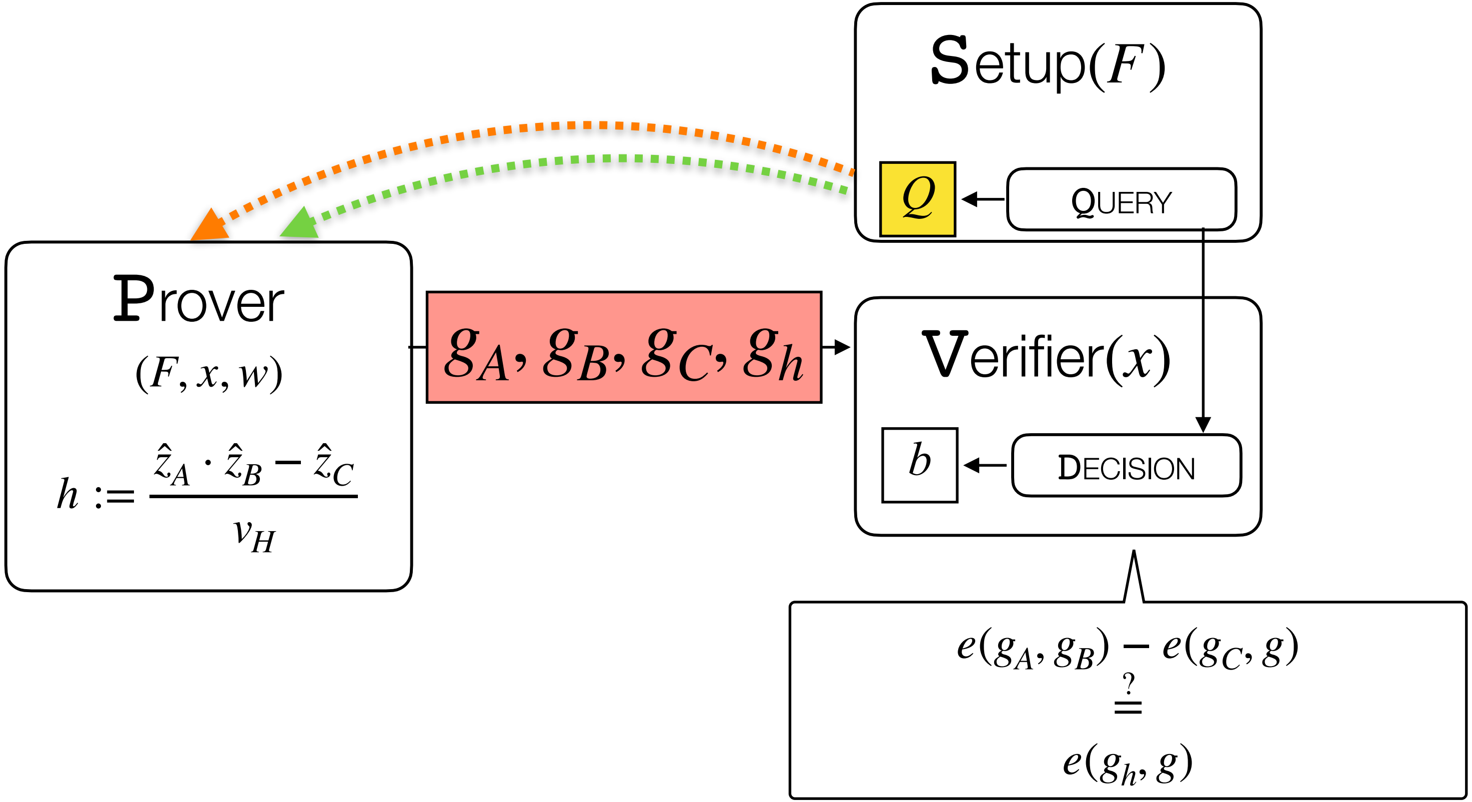
$$g_C := g^{\langle \overrightarrow{L(r)} C, z \rangle}$$

$$g_h := g^{h(r) v_H(r)}$$

We need to check

$$\langle \overrightarrow{L(r)} A, z \rangle \cdot \langle \overrightarrow{L(r)} B, z \rangle - \langle \overrightarrow{L(r)} C, z \rangle \stackrel{?}{=} h(r) v_H(r)$$

Decision via pairing



Let's analyze this: Soundness

Assuming GGM, malicious prover can only compute linear combinations of pk

So it *must* provide a linear response to encoded queries

Additionally, DL ensures that prover learns nothing about query.

Summary

- Proof size: 4 group elements
- Setup work: $O(n \log n)\ \mathbb{F} + O(n)\ \mathbb{G}$
- Prover work: $O(n \log n)\ \mathbb{F} + O(n)\ \mathbb{G}$
- Verifier work: 3 pairings

Unresolved questions:

- What about public input?
- What if prover uses different oracles for A, B, C?
- What if prover's response is *affine*?