Efficient Cryptographic Proofs from RAA Codes

Martijn Brehm
University of Amsterdam

Binyi Chen
Stanford University

Ben Fisch
Yale University

Nicolas Resch
University of Amsterdam

Ron D. Rothblum
Succinct

Hadas Zeilberger
Yale University

Succinct Non-Interactive Arguments (SNARGs)

[Kilian'92, Micali'94]

Prover: (x, w)

proof π

Verifier: x

Key bottleneck in practice

Targets:

- *Succinct* proof: $|\pi| \ll |w|$
- *Efficient* prover (linear time)
- *Efficient* verifier (polylog time)

Soundness against *poly-time* provers

$L = \{x : \exists w, \Gamma(x, w) = 1\}$

Recall: Interactive Oracle Proofs (IOPs)

[Reingold-Rothblum-Rothblum'16, BenSasson-Chiesa-Spooner'16]

Hybrid between (public-coin) interactive proofs and PCPs

Prover: (x, w)

Verifier: x

queries $\ll |w|$

proof π_1 →

← challenge r_1

proof π_2 →

← challenge r_2

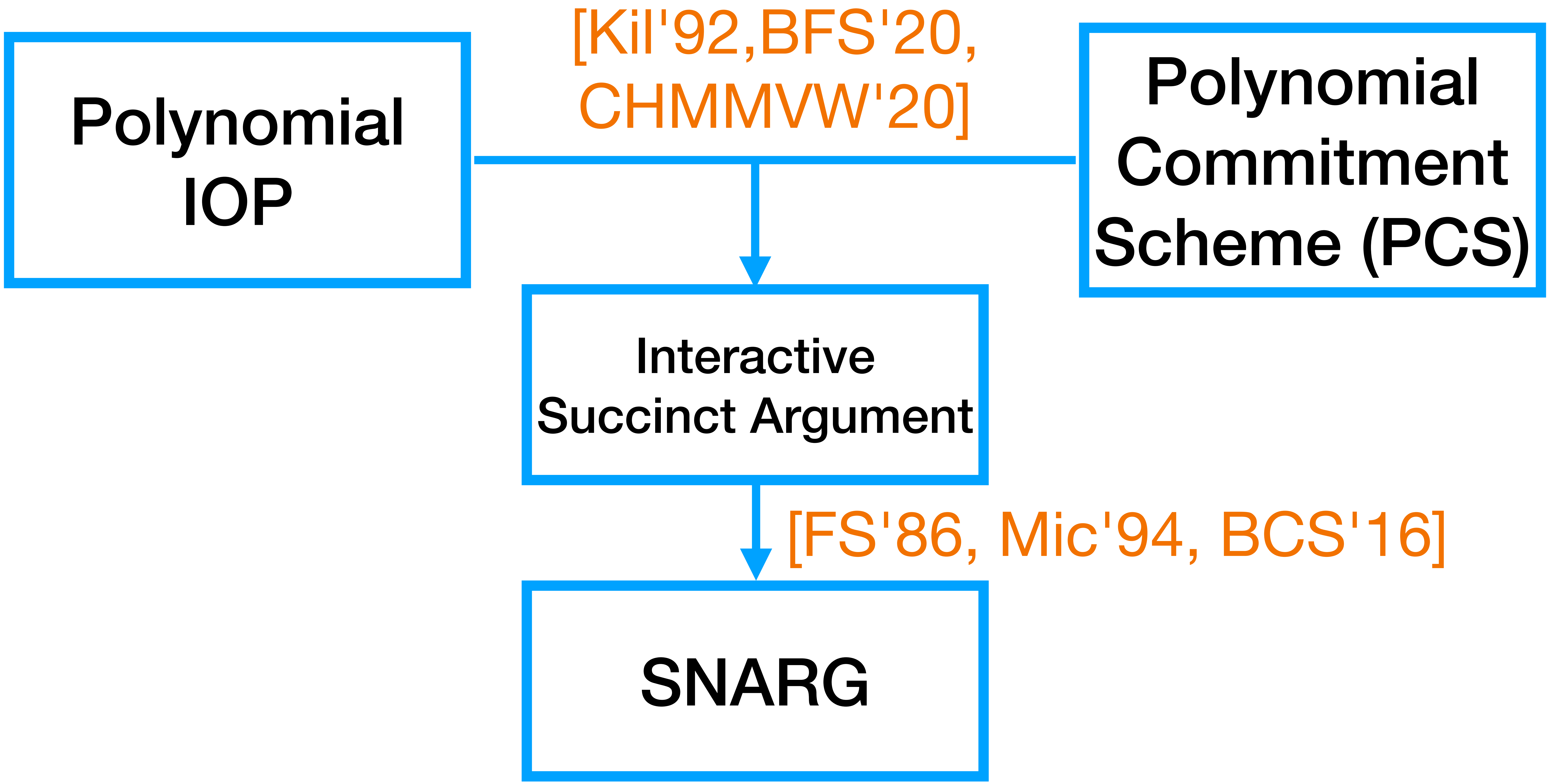
⋮

$x \overset{?}{\in} L$

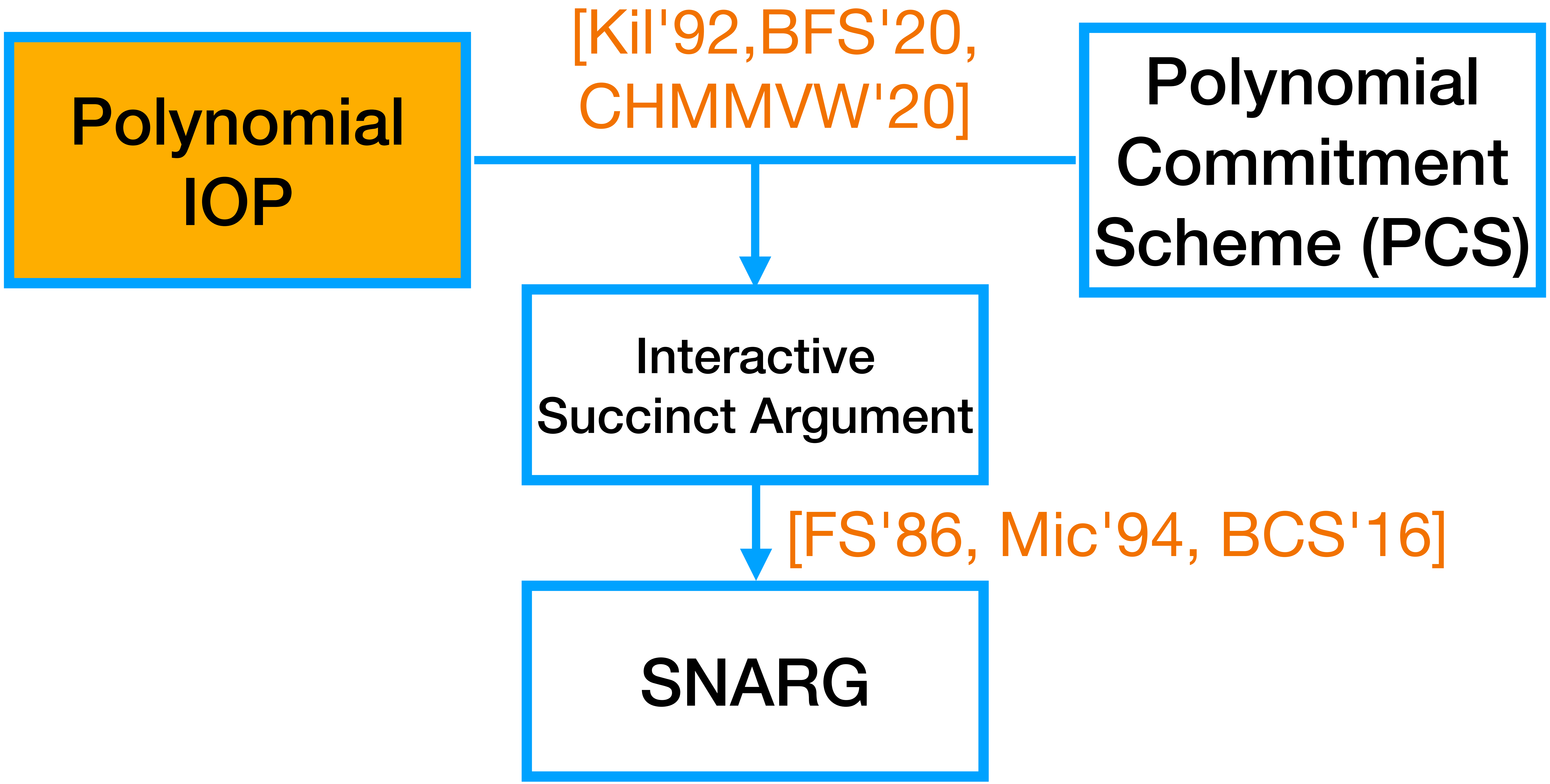
$L = \{x : \exists w, \Gamma(x, w) = 1\}$

Soundness against *arbitrary* provers

Common SNARG Construction Paradigm



Common SNARG Construction Paradigm



Polynomial IOPs

[RRR'16, BFS'20, CHMMVW'20]

Prover: (x, w)

Verifier: x

(evaluations of) polynomial P_1 →

← challenge r_1

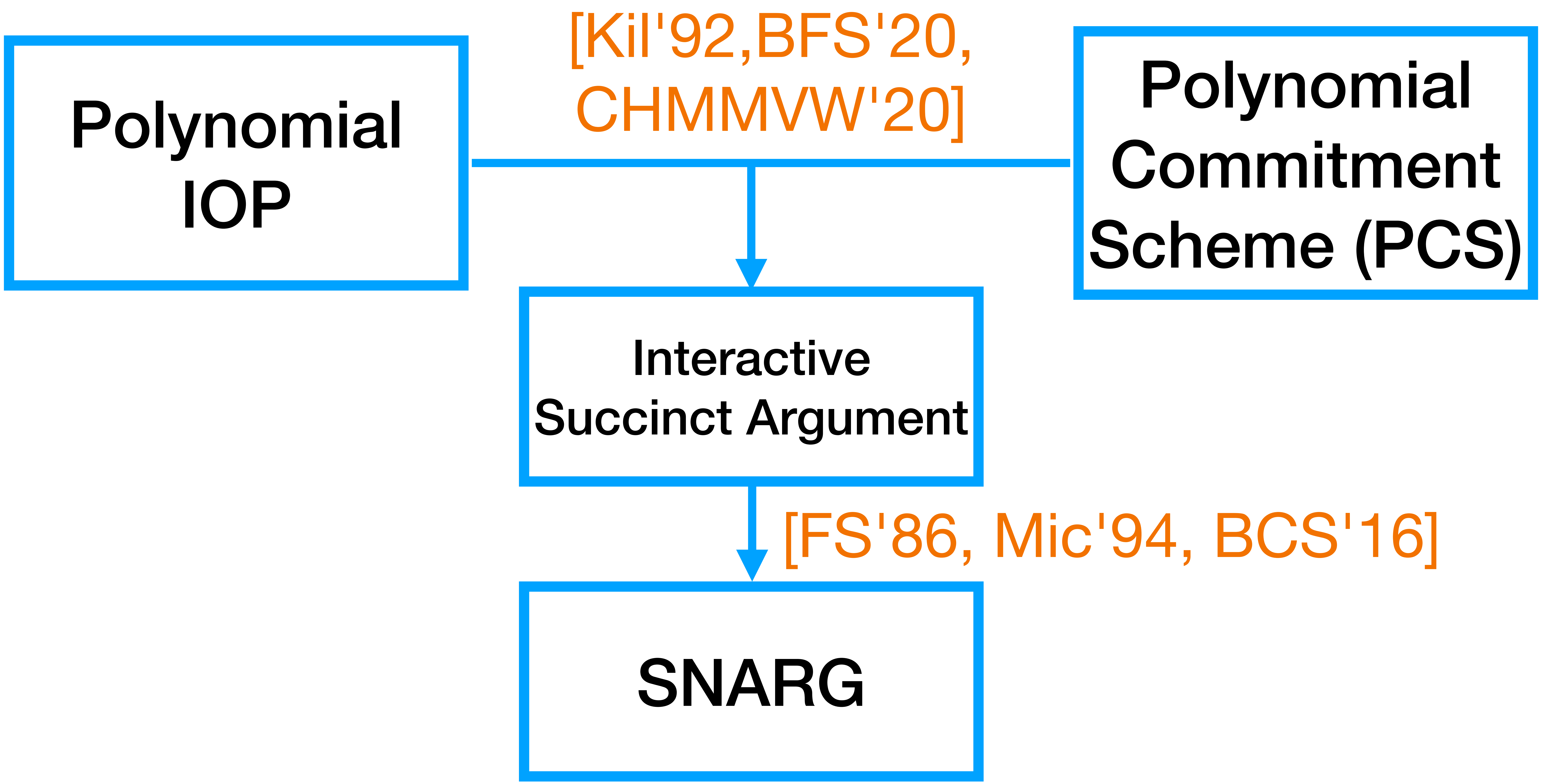
(evaluations of) polynomial P_2 →

← challenge r_2

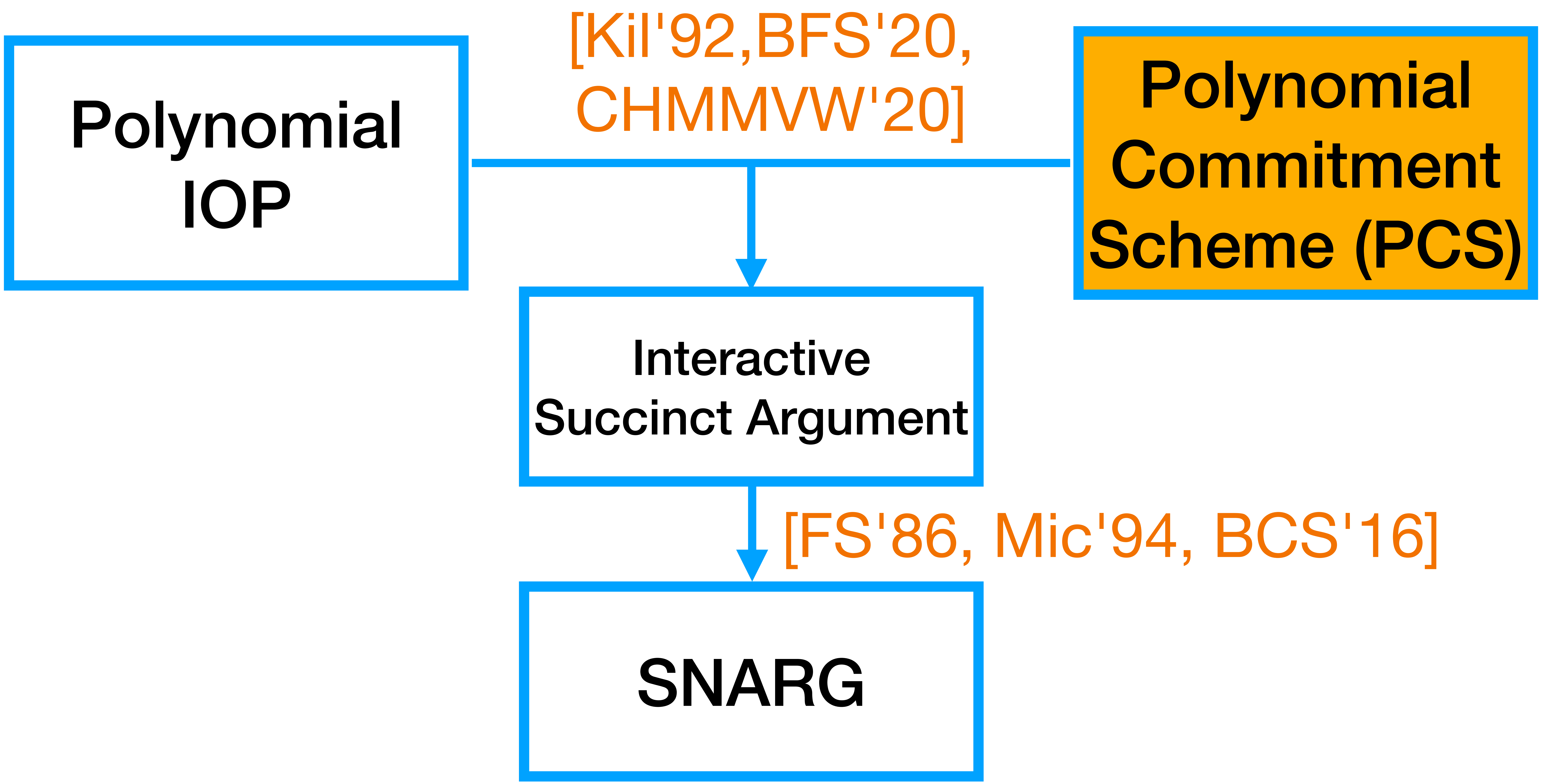
⋮

Queries:
what is $P_i(\alpha)$?

Common SNARG Construction Paradigm

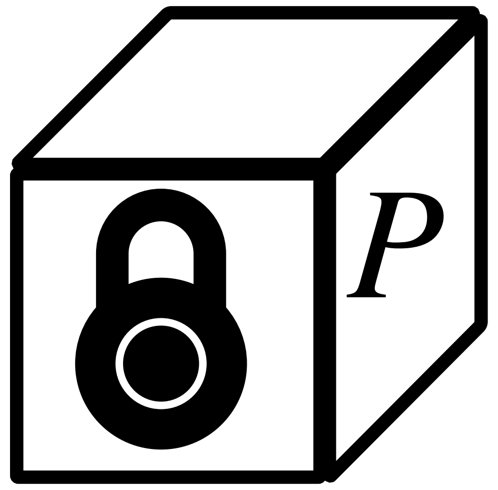


Common SNARG Construction Paradigm



Polynomial Commitment Scheme (PCS)

Key bottleneck in modern SNARGs



(short) commitment to P

I'm convinced $P(x) = y$

Later...

what is value of P on x?

it is y

Blaze: a new fast *Multilinear* PCS (MLPCS)

- Focus on committing to multilinear polynomials over characteristic 2 fields

- That is, $P : \mathbb{F}_{2^\lambda}^r \to \mathbb{F}_{2^\lambda}$ is of the form

$$P(X_1, \ldots, X_r) = \sum_{S \subseteq [r]} c_S \prod_{i \in S} X_i, \qquad c_S \in \mathbb{F}_{2^\lambda}$$

- Go hand-in-hand with highly efficient multilinear polynomial IOPs: Spartan [Set'20], Lasso/Jolt [STW'24, AST'24], Hyperplonk [CBBS'23] and generally GKR-based schemes such as Orion [XZS'22]

Why Characteristic 2?

Pros

- Addition is fast
- Eliminate "embedding overhead" [DP23]
- Friendly to arithmetization of logical operations

Cons

- Multiplication is a bit more complicated (but we won't be doing much!)
- Less friendly to integer arithmetic

Real reason: we use a code that we currently only know how to analyze over $\mathbb{F}_2$…

Blaze Asymptotics

- Cost of committing to $P : \mathbb{F}^r \to \mathbb{F}$ (described by $N = 2^r$ field elements)
- Commitment generation: $8N$ additions + 1 Merkle Hash
- Evaluation proof generation: $6N$ additions + $5N$ multiplications
- Proof length anc $\mathrm{N})^2\big)$

Prior works:

$O(N \log N)$, or $O(N)$

with unspecified constant

Techniques: Bird's Eye View

1. Building on code-switching technique [RonZewi-Ron'20], using *code interleaving* we build an MLPCS from

   MLPCS for "smaller" polynomials

   Leads to proof-size & verification time

   Error-correcting code

   Leads to proving time & committing time

2. Use Repeat-Accumulate-Accumulate (RAA) codes, which have *very* fast encoding and (usually) good distance

The Multi-Linear Polynomial Commitment Scheme (MLPCS)

Multilinear Extension

Will identify $u \in \mathbb{F}^k$ with a function $u : \{0,1\}^{\log k} \to \mathbb{F}$

- Given a function $f : \{0,1\}^r \to \mathbb{F}$, there exists a unique $\hat{f} : \mathbb{F}^r \to \mathbb{F}$ s.t.

$\hat{f}$ is multilinear

$\hat{f}(x) = f(x)$ for all $x \in \{0,1\}^r$

- Specifically, take

$$\hat{f}(x) = \sum_{b \in \{0,1\}^r} f(b) \cdot \mathrm{eq}(b, x)$$

where $\mathrm{eq}(b, x) = \prod_{i=1}^{r} \left(b_i x_i + (1 - b_i)(1 - x_i)\right)$

MLPCS from Codes & IOPP

$(k = 2^r)$

Fix a code $\mathscr{C} : \mathbb{F}^k \to \mathbb{F}^n$

To get a multilinear polynomial commitment scheme (MLPCS),

suffices to design an interactive oracle proof of proximity (IOPP) for the language [FS'86, Kil'92, Mic'94, BCS'16]

$$\{\mathscr{C}(m) \in \mathbb{F}^n : m \in \mathbb{F}^k \text{ and } \hat{m}(z) = v\}$$

only need to reject if input is far (in Hamming distance) from language

Interleaved Codes

$t = \Theta(\log n)$

Identify $M \in \mathbb{F}^{t \times k}$, $M \in \mathbb{F}^{tk}$ and $M : \{0,1\}^{\log t + \log k} \to \mathbb{F}$

Given $\mathscr{C} : \mathbb{F}^k \to \mathbb{F}^n$ and integer t

Construct new code $\mathscr{C}^t : \mathbb{F}^{tk} \to \mathbb{F}^{tn}$ by interleaving [Ligero, BCGGHJ'17]

Assuming IOPP for $\{\mathscr{C}(m) \in \mathbb{F}^n : m \in \mathbb{F}^k$ and $\hat{m}(z) = v\}$ exists,

we construct IOPP for $\{\mathscr{C}^t(M) : M \in \mathbb{F}^{t \times k}, \hat{M}(z) = v\}$ $(z \in \mathbb{F}^{\log(tk)})$

$M \in \mathbb{F}^{t \times k}, v \in \mathbb{F}$
$z \in \mathbb{F}^{\log(t)+\log(k)}$

$$C = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{bmatrix}$$

$v \in \mathbb{F},$
$z \in \mathbb{F}^{\log(t)+\log(k)}$

Idea: decompose to claim on *rows*

Need to prove
$\hat{M}(z) = v$

$$\hat{M}(z) = \sum_{b \in \{0,1\}^{\log t + \log k}} \mathrm{eq}(b, z) \cdot M(b)$$
$$= \sum_{b_1 \in \{0,1\}^{\log t},\ b_2 \in \{0,1\}^{\log k}} \mathrm{eq}(b_1, z_1) \cdot \mathrm{eq}(b_2, z_2) \cdot M(b_1, b_2)$$
$$= \sum_{b_1 \in \{0,1\}^{\log t}} \mathrm{eq}(b_1, z_1) \cdot \hat{M}_{b_1}(z_2)$$

$z = (z_1, z_2) \in \mathbb{F}^{\log t} \times \mathbb{F}^{\log k}$

Multilinear Evaluation with Interleaving

1. Prover sends function $u : \{0,1\}^{\log t} \to \mathbb{F}$ defined as

$$u(x) = \hat{M}_x(z_2)$$

Send random linear combination of $m_1, \ldots, m_t$

2. Verifier checks that

$$\hat{u}(z_1) = \sum_{b_1 \in \{0,1\}^{\log t}} \mathrm{eq}(b_1, z_1) \cdot u(b_1) = v$$

If Verifier check passes but $\hat{M}(z) \neq v$, then Prover sent incorrect u

How to enforce correctness?

$M \in \mathbb{F}^{t \times k}, v \in \mathbb{F}$
$z \in \mathbb{F}^{\log(t)+\log(k)}$

$C = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_t \end{bmatrix}$

$v \in \mathbb{F},$
$z \in \mathbb{F}^{\log(t)+\log(k)}$

$\xrightarrow{u}$

$\xleftarrow{r}$

$\xrightarrow{c_0}$

Compute $u(x) = \hat{M}_x(z_2)$

Check $\hat{u}(z_1) \stackrel{?}{=} v$

Sample $r \in \mathbb{F}^t$

$c_0 := \Sigma_i\ r_i c_i$, which is $\mathscr{C}$-encoding of $m_0 := \Sigma_i\ r_i m_i$

Run IOPP for $\mathscr{C}$ with $m_0, c_0, z_2, v_0 := \Sigma_i\ r_i u_i$

Check consistency of c_0 and C for $\Omega(\lambda/\delta)$ $i \in [n]$

Proximity Gaps

What if $C \in \mathbb{F}^{t \times n}$ is far from $\mathscr{C}^t$?

Can use elegant results on *proximity gaps* for codes [RVW'13, BKS'18] to guarantee that whp, $c_0 = \Sigma_i\ r_i c_i$ is far from $\mathscr{C}$

Theorem: ([BKS'18]) Suppose $\mathscr{C} \leq \mathbb{F}^n$ has min. distance δ, let $U \subseteq \mathbb{F}^n$ be an affine space and suppose $\exists u \in U$ for which $\Delta(u, \mathscr{C}) > \tau$. Then if $\varepsilon > 0$ is s.t. $\tau - \varepsilon < \delta/3$,

$$\mathbb{P}_{u \in U}[\Delta(u, \mathscr{C}) < \tau - \varepsilon] \leq (\varepsilon \mathbb{F})^{-1}$$

Summary

Prover time dominated by computation of t $\mathscr{C}$-encodings

So: pick code with *blazing fast* encoding!

Additionally: for soundness need good distance

can use "off-the-shelf" constructions

Actually, $\mathscr{C}$ better already have an IOPP for multilinear evaluation…

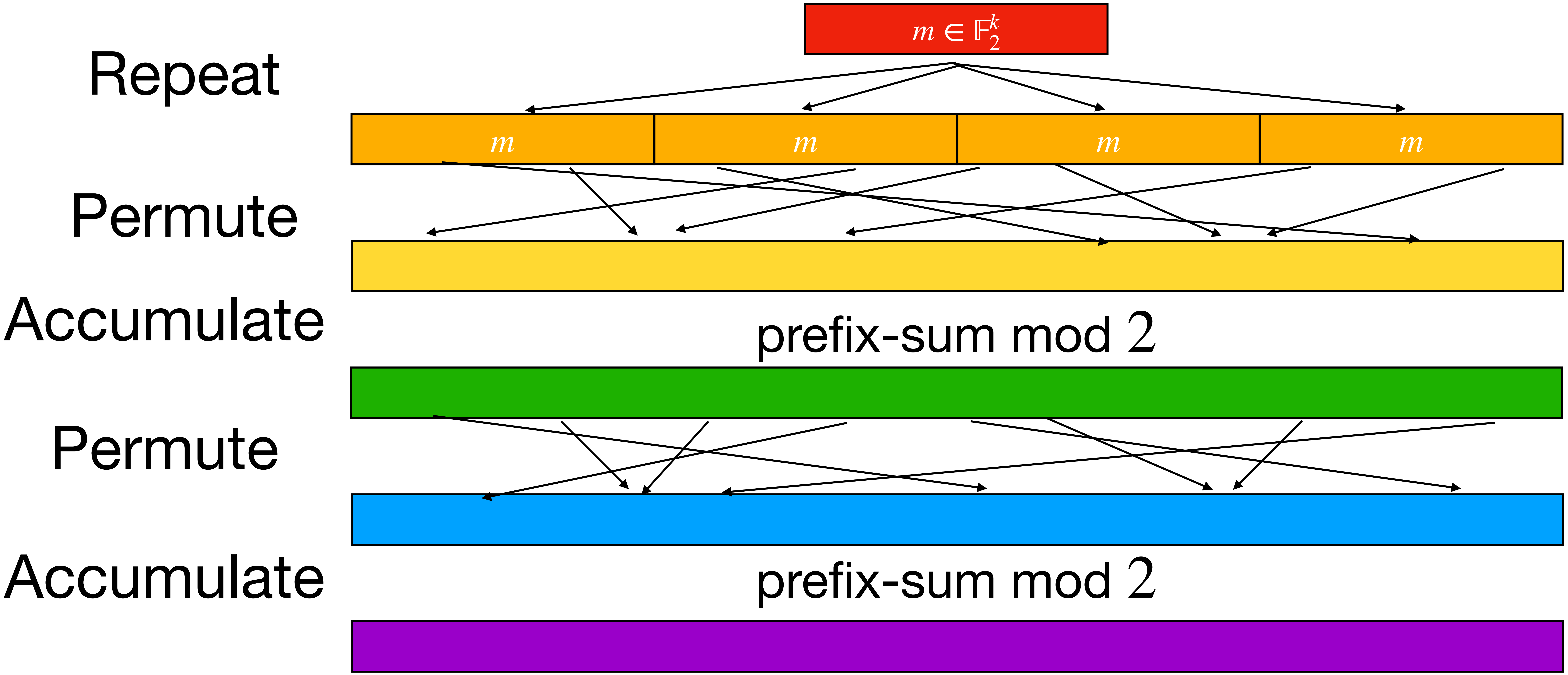
but we actually provide tailor-made one for our choice of $\mathscr{C}$

Repeat-Accumulate-Accumulate Codes [Divsalar-Jin-McEliece'98]

RAA Encoding

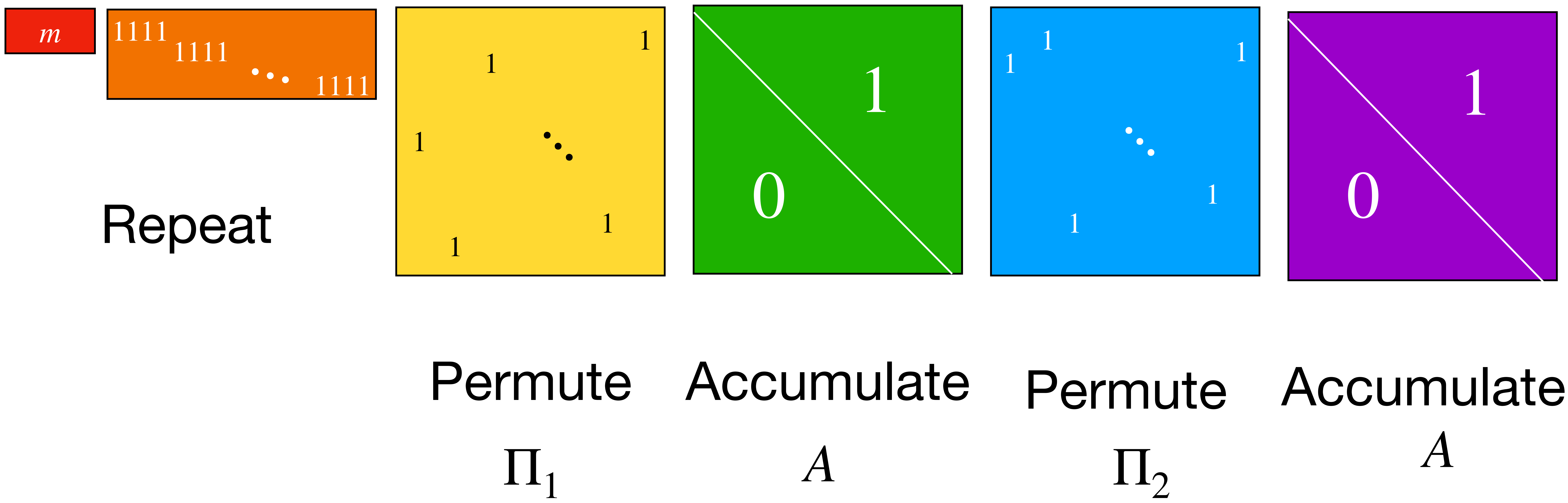
working over $\mathbb{F}_2$...

(rate $1/4$ case)



RAA Generator Matrix

(rate $1/4$ case)



Analysis

Over random choice of 2 permutations, we show whp

a rate $1/4$ RAA code has min. distance ≥ 0.19

GV bound for rate $1/4$: ~ 0.21

This builds off prior works [PS'03, BMS'07, KZKC'07, RF'09]

Use *Input-Output-Weight-Enumerator Function*

$$N(a,b) := |\{x \in \mathbb{F}_2^n : \mathrm{wt}(x) = a \text{ and } \mathrm{wt}(xA) = b\}|$$

$$= \binom{b-1}{\lceil a/2 \rceil - 1} \cdot \binom{n-b}{\lfloor a/2 \rfloor}$$

$$p_{a \to b} := \mathop{\mathbb{P}}_{\substack{x \in \mathbb{F}_2^n, \\ \mathrm{wt}(x) = a}} \left[\mathrm{wt}(xA) = b\right] = \frac{N(a,b)}{\binom{n}{a}}$$

Analysis

Let $X =$ number of codewords of weight $\leq d$

By Markov: $\mathbb{P}[\mathrm{dist}(\mathscr{C}) \leq d] = \mathbb{P}[X \geq 1] \leq \mathbb{E}[X]$

number of messages

prob. of going from *repeated* message weight to intermediate weight

$$\mathbb{E}[X] = \sum_{a=1}^{n/4} \sum_{b=1}^{n} \sum_{w=1}^{d} \binom{n/4}{a} \cdot p_{4a \to b} \cdot p_{b \to w}$$

choices for message weight

choices for intermediate weight

choices for codeword weight

prob. of going from intermediate weight to codeword weight

Analysis

$$\mathbb{E}[X] = \sum_{a=1}^{n/4} \sum_{b=1}^{n} \sum_{w=1}^{d} \binom{n/4}{a} \cdot p_{4a\to b} \cdot p_{b \to w}$$

Break up sum based on b

$b \le h = \omega(\log n)$:

use $\binom{a}{b} \le \left(\frac{ea}{b}\right)^b$;

final bound $1/n^\varepsilon$

$b \ge h = \omega(\log n)$:

use $\binom{a}{b} \le 2^{aH(b/a)}$;

final bound $2^{-\Omega(h)}$

Tests

From crypto perspective: failure probability $1/n^{\varepsilon}$ quite large…

also, $1/n^{O(1)}$ unavoidable (consider prob. of wei
$O(1) \to O(1) \to O(1)$; all terms $1/\mathrm{poly}(n)$)

Actually, just check first round accumulation

To boost failure probability: check encoding of low-weight messages

Conditioned on test passing, failure probability decreases substantially

but with poly-time test, failure probability still $\geq 1/\mathrm{poly}(n)$…

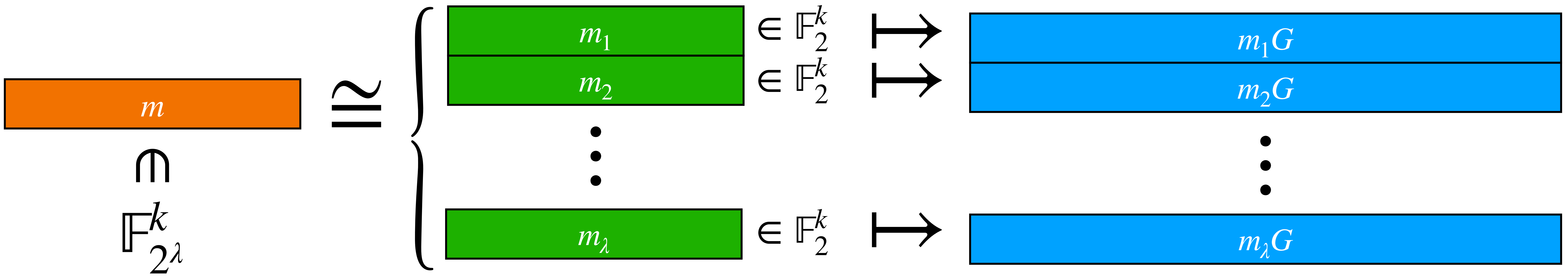
RAA over Extension Fields

Thus far: analyzed RAA codes over $\mathbb{F}_2$

But: for soundness, need $|\mathbb{F}| = 2^\lambda, \lambda \approx 128$

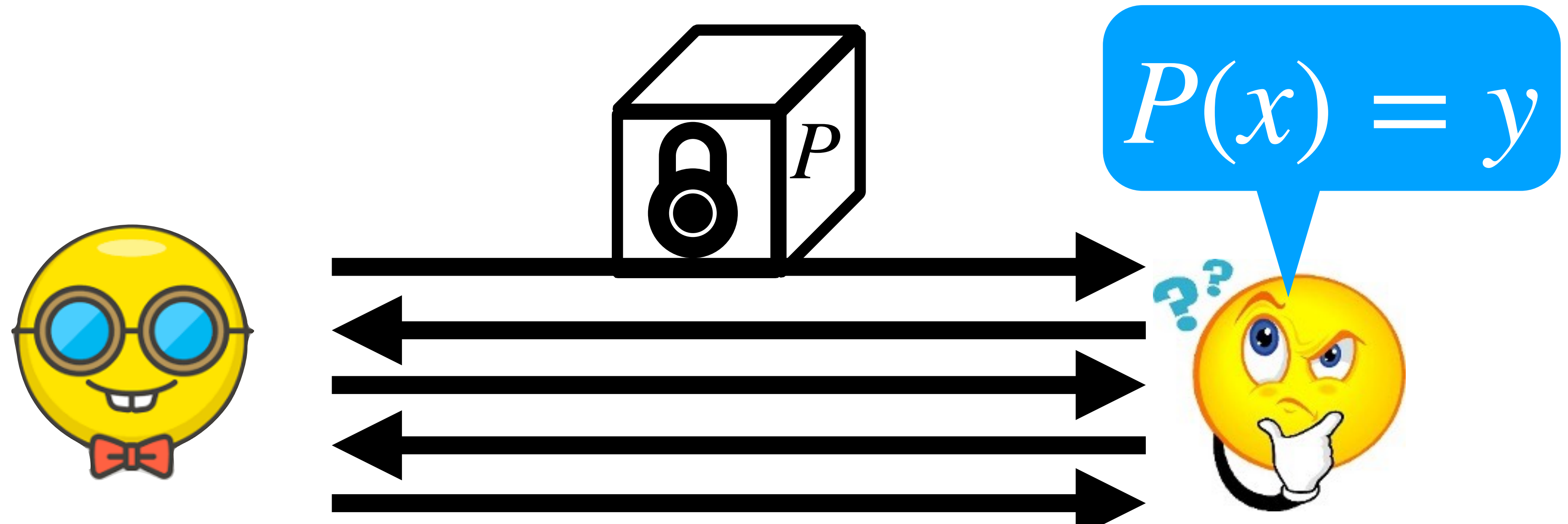
Can use same generator matrix G to define code over $\mathbb{F}_{2^\lambda}$

Can implement by "bit-slicing"



Conclusion

Recap



1. Building on code-switching technique [RonZewi-Ron'20], using *code interleaving* we build an MLPCS from

"Smaller" MLPCS

Leads to proof-size & verification time

$\Longleftarrow$ IOPP for $\{\mathscr{C}(m) \in \mathbb{F}^n : m \in \mathbb{F}^k$ and $\hat{m}(z) = v\}$

Error-correcting code

Leads to proving time & committing time

2. Use Repeat-Accumulate-Accumulate (RAA) codes, which have *very* fast encoding and are (usually) near GV-bound

Open Problems

- Analyze RA* codes directly over larger alphabets? [BFKTWZ'24] Can we approach the Singleton bound?
- (More) explicit constructions? Like [Applebaum-Kachlon'20], design test that leads to negligible $n^{-\omega(1)}$ failure probability?
- Could puncturing be useful?
- Concrete bounds for smaller n? (currently need $n \approx 2^{20}$)
- Improved proximity gaps? Maybe tailor-made for RAA codes?

Thank you!