BaseFold: Fast Field-agnostic PCS from Foldable Codes

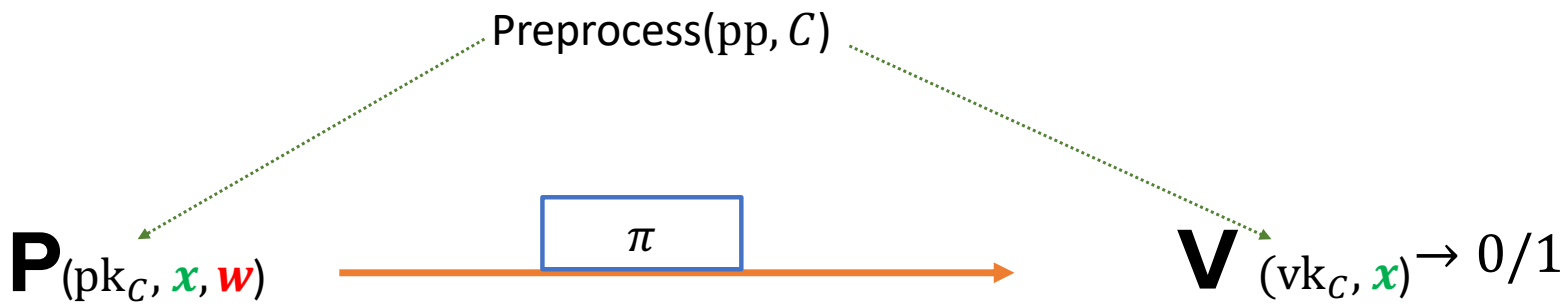
Hadas Zeilberger
Yale

Binyi Chen
Stanford

Ben Fisch
Yale

(zk)SNARKs

(zk)SNARK = A *succinct* ZK proof showing that $\exists \boldsymbol{w}$ s.t. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$



Requirements for π:

- **Completeness**: honest P can compute valid π
- **Knowledge soundness**: P* knows valid $\boldsymbol{w}$ if it can generate valid π
- **Succinctness**: $|\pi| \ll |\boldsymbol{w}|$ and fast to verify
- **Zero knowledge**: π hide the witness $\boldsymbol{w}$

(zk)SNARKs

(zk)SNARK = A *succinct* ZK proof showing that $\exists \boldsymbol{w}$ s.t. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$

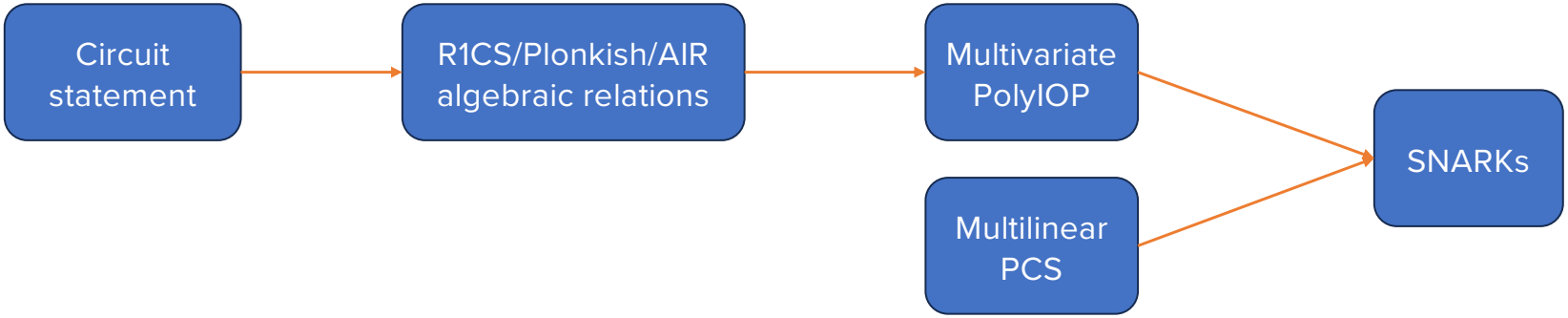
Tons of applications:

- ZK-Rollups/zkEVMs
- Verifiable Machine learning [ZEN, zkCNN, zkDT, Otti, etc...]
- Fighting image disinformation [NT16, BD22, KHSS22]
- Privacy-preserving smart-contracts/payment systems [Zcash, Zexe, Veri-ZEXE, etc...]

Key performance demands:

- Proving as fast as computing
  - Efficient transform from circuit statements to algebraic relations
  - Fast (e.g. linear-time) proving algorithm for the algebraic relations
- *Small* (e.g., logarithmic) proof size/verification cost for *large* circuits

Multivariate-PolyIOP-based SNARKs



Adapt [BFS19, CHMMVW19] to the multivariate setting

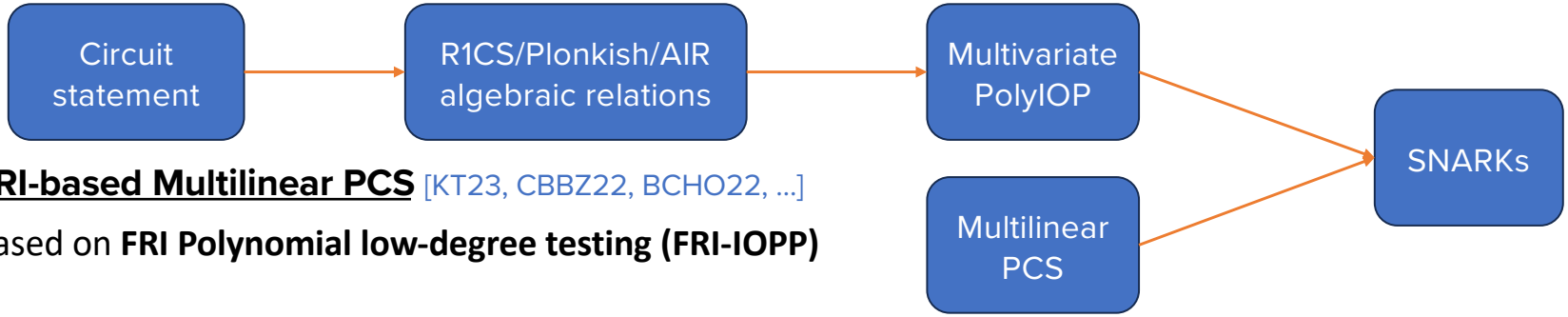
Efficiency bottleneck – polynomial commitments:

- DLOG/Pairing-based PCS [KZG10/PST13, Bulletproofs, …]:
  - expensive ECC operations
- PCS from tensor-based IOPPs [Brakedown, Ligero, BCG20, …]:
  - expensive proof size/verification time

Why multivariate PolyIOP? [CBBZ22, Setty19, etc]

- Prover performs a linear # of field ops
- No restriction on the field choice

Multivariate-PolyIOP-based SNARKs



FRI-based Multilinear PCS [KT23, CBBZ22, BCHO22, …]

Based on **FRI Polynomial low-degree testing (FRI-IOPP)**

Adapt [BFS19, CHMMVW19] to the multivariate setting

Advantages:

- No expensive **cryptographic** ops
- Transparent and plausibly post-quantum secure
- **Polylogarithmic** proof size/verification

Disadvantages:

- Require **FFT-friendly** fields
- Extra uni-to-multivariate PCS compilation overhead

ECFFT[BCKL22] supports general fields but not concretely efficient

Efficient Multilinear PCS

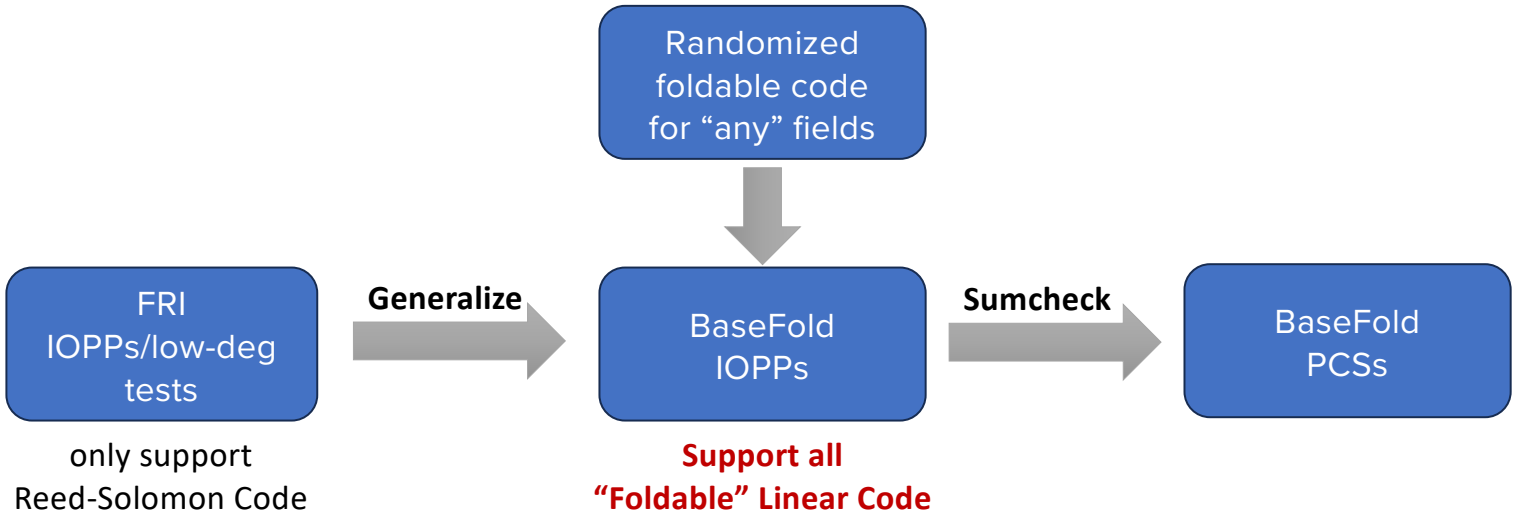
Question: Can we
(i) have a **field-agnostic** FRI-like IOPP, and
(ii) directly compile the IOPP to a multilinear PCS?

Why Field-Agnosticity?

Avoid **non-native** field simulation in the circuit
(e.g., ECDSA signature verification circuit)

Can use the field with the most efficient
CPU/hardware implementations
(e.g., Mersenne field, BabyBear field)

BaseFold: A new multilinear PCS



BaseFold Multilinear PCS:

- **Support "arbitrary" fields!**
- No uni-to-multivariate PCS compilation overhead
- Preserves other efficiency advantages of FRI IOPP

Related work:

- [ABN23]: IOPP for polynomial codes
- [BLNR22]: IOPP to algebraic geometry codes

Agenda

- IOPP from “foldable” linear codes
- Foldable linear codes over fast fields
- Efficient multilinear PCS compiler from BaseFold IOPPs

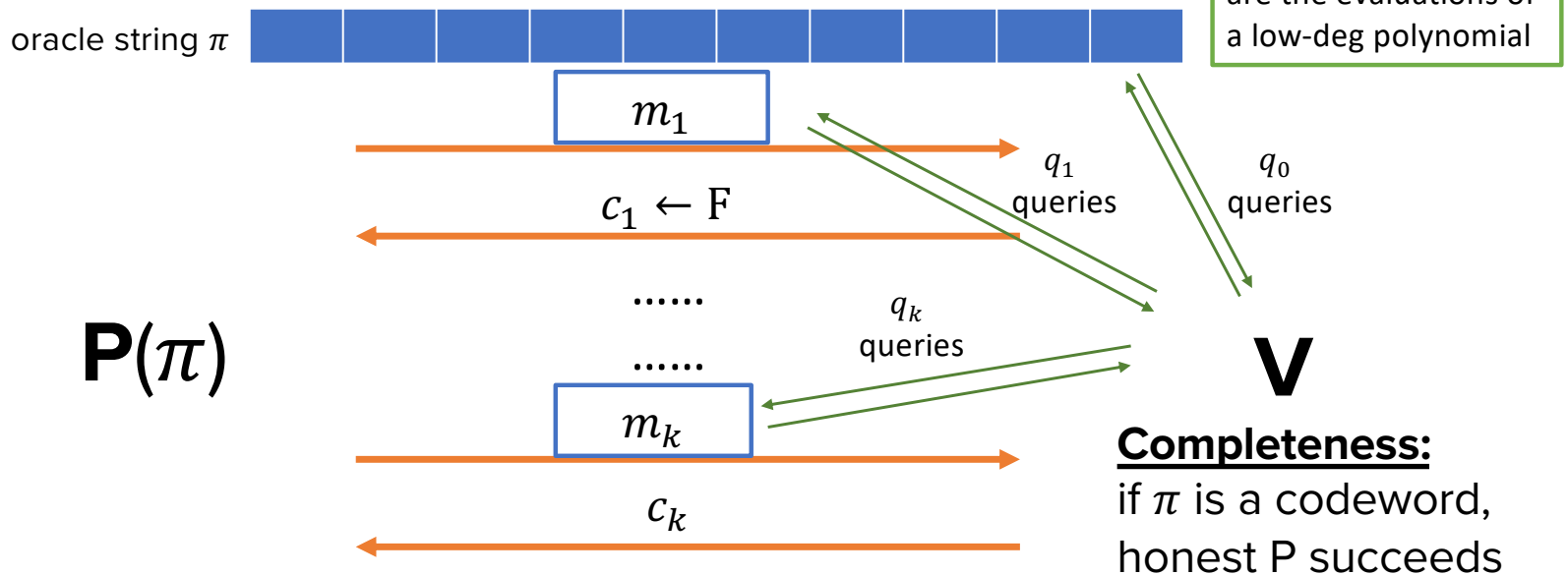
Agenda

- IOPP from “foldable” linear codes
- Foldable linear codes over fast fields
- Efficient multilinear PCS compiler from BaseFold IOPPs

IOPPs

Goal: check π is close to a codeword

E.g., checking π are the evaluations of a low-deg polynomial



Completeness:
if π is a codeword, honest P succeeds

Soundness:
if π far from codewords, V rejects w.h.p.

Complexity:
- P/V time
- $|q_0| + \cdots + |q_k|$

FRI IOPPs

The evals $f_{|L}$ of oracle poly $f(X)$ in domain $L \subseteq \mathbb{F}$
E.g., $L = \{2^n\text{-th root-of-unities}\}$

Goal:
check $\deg(f) \leq 2^k$

$f(X) = f_e(X^2) + Xf_o(X^2)$

$c_1 \leftarrow \mathbb{F}$

$f^{(1)}(X) = f_e(X) + c_1 f_o(X)$

$f^{(1)}_{|L_1}$ L_1 = "half" of L $\deg(f^{(1)}) \leq 2^{k-1}$

P(π) ..... **V**

c_k

$f^{(k-1)}(X) = f_e^{(k)}(X^2) + Xf_o^{(k)}(X^2)$

$f^{(k)}(X) = f_e^{(k)}(X) + c_k f_o^{(k)}(X)$

$f^{(k)}_{|L_k}$

V query checks:

$\mathbb{F}$ **has to be FFT-friendly : (**

1. $f^{(k)}$ is a const poly
2. Sample random $u^{(1)} \leftarrow_\$ L$ and derive $u^{(2)}, \ldots, u^{(k)}$ from $u^{(1)}$
3. $\forall i \in [k], f^{(i)}$ consistent w/ $f^{(i-1)}$:
   - $f^{(i)}(u^{(i)}) = f_e^{(i)}(u^{(i)}) + c_i f_o^{(i)}(u^{(i)})$ on $u^{(i)} \in L_i$

e.g., $f_e^{(1)}(u)$ obtained/interpolated given $f_{|L}(\pm\sqrt{u})$

Question: Can we generalize beyond RS Code?

Foldable Linear Codes

“Foldable” Linear Codes:

- A family of *linear* codes with encoding algorithms $\left\{\mathrm{Enc}_i\colon \mathbb{F}^{2^i} \to \mathbb{F}^{c2^i}\right\}_{i=1}^{d}$
- Parameterized by a list of vectors $\{\vec{t}_{i,L}, \vec{t}_{i,R} \in \mathbb{F}^{c2^i}\}_{i\in[d]}$ s.t.$\forall j$: $\vec{t}_{i,L}[j] \neq \vec{t}_{i,R}[j]$

“Foldability” :

$$\mathrm{Enc}_{i+1}(\vec{m}_L||\vec{m}_R) = \left(\begin{array}{c} \mathrm{Enc}_i(\vec{m}_L) \\ + \\ \vec{t}_{i,L} \\ \circ \\ \mathrm{Enc}_i(\vec{m}_R) \end{array} \;\Big|\Big|\; \begin{array}{c} \mathrm{Enc}_i(\vec{m}_L) \\ + \\ \vec{t}_{i,R} \\ \circ \\ \mathrm{Enc}_i(\vec{m}_R) \end{array} \right)$$

Examples:
Reed-Solomon code, etc...

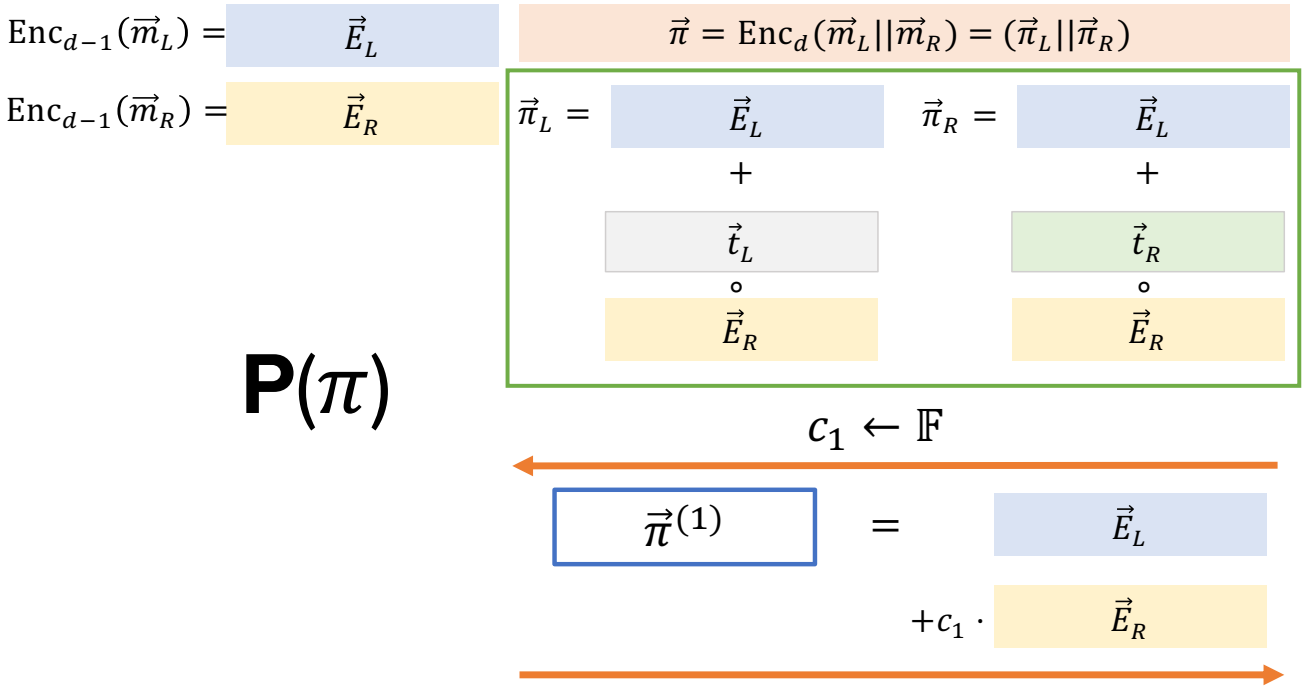
Question: How to generalize FRI for **any** foldable codes?

$\mathrm{Enc}_i(\vec{m}_L)$ & $\mathrm{Enc}_i(\vec{m}_R)$: coef form of the $c2^i$ linear polynomials

$\mathrm{Enc}_{i+1}(\vec{m}_L||\vec{m}_R)$: point eval form of $c2^i$ linear polynomials:

- The t-vectors are the evaluation points

IOPPs from any Foldable Codes



Goal:
check π "close" to a codeword in C_d

Consistency check:

- $\vec{\pi}^{(1)}[u^{(1)}] = \vec{E}_L[u^{(1)}] + c_1\vec{E}_R[u^{(1)}]$ for rand entries $u^{(1)}$
- $\vec{E}_{L/R}[u]$ can be interpolated given $\vec{\pi}_{L/R}[u]$

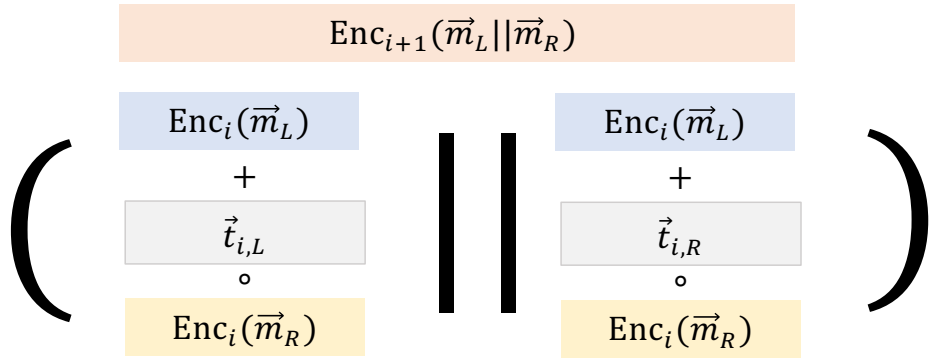
By linearity, $\pi^{(1)} = \mathrm{Enc}_{d-1}(\vec{m}_L + c_1\vec{m}_R)$, "close" to C_{d-1}

Agenda

- IOPP from “foldable” linear codes
- Randomized foldable linear codes over fast fields
- Efficient multilinear PCS compiler from BaseFold IOPPs

Randomized Foldable Code

Recall "Foldable" Linear Codes:



Observation:
The code family is fixed after fixing Enc_0 and the t-vectors $\{\vec{t}_{i,L}, \vec{t}_{i,R}\}_{i=0}^{d}$

Question:
How to set $\vec{t}_L$ and $\vec{t}_R$ for
- Good min-distance
- Field agnosticism

Our construction:
- $\vec{t}_{i,L} \leftarrow_{\$} (\mathbb{F}^{\times})^{c2^i}, \vec{t}_{i,R} := -\vec{t}_{i,L}$
- Almost no field restrictions! (except $|\mathbb{F}| > 1000$)
- Trivial recursive encoding:
  - $O(n \log n)$ F-ops w/ super small constant

Key Question:
Why does it have good minimum relative distances?

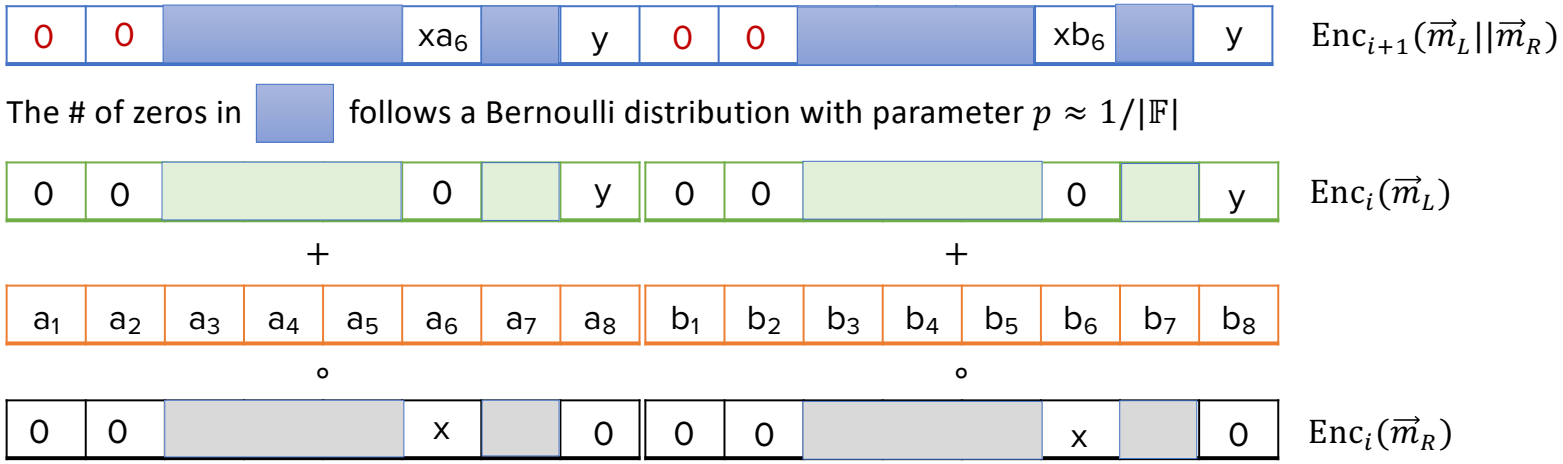
Minimum Distance Analysis

Enc_i has $< L$ zeros ⇒
$\text{Enc}_{i+1}(\vec{m}_L||\vec{m}_R)$ has $\leq 2L$ red zeros

$\text{Enc}_{i+1}(\vec{m}_L||\vec{m}_R)$ has
$\geq 2L + \delta$ zeros w.p. $\sim \frac{2^{c|\vec{m}_L|}}{|\mathbb{F}|^{\delta}}$

Goal: Prove $\text{Enc}_i(\vec{m})$ has very few zeros for any $\vec{m} \neq \vec{0}$

Minimum Distance = Minimum (nonzero) Hamming Weight



Challenge: Need take union bound over all $|\mathbb{F}|^{2|\vec{m}_L|} - 1$ messages ⇒ large δ : (

Minimum Distance Analysis

Goal: Prove $\mathrm{Enc}_i(\vec{m})$ has very few zeros for any $\vec{m} \neq \vec{0}$

Minimum Distance = Minimum (nonzero) Hamming Weight

$\mathrm{Enc}_{i+1}(\vec{m}_L||\vec{m}_R)$

The # of zeros in ▭ follows a Bernoulli distribution with parameter $p \approx 1/|\mathbb{F}|$

fewer red zeros

Observation: For many $\vec{m} = \vec{m}_L||\vec{m}_R$, $\mathrm{Enc}_i(\vec{m}_L)$ and $\mathrm{Enc}_i(\vec{m}_R)$ have $\ll L$ zeros

For any $\vec{m} = \vec{m}_L||\vec{m}_R$, denote S as the **maximal** subset s.t. $\mathrm{Enc}_i(\vec{m}_L)[S] = \mathrm{Enc}_i(\vec{m}_R)[S] = 0^{|S|}$

$$\Pr[\mathrm{Enc}_{i+1}(\vec{m}_L||\vec{m}_R) \text{ has } 2L + \delta \text{ zeros}] \sim \frac{2^{c|\vec{m}_L|}}{|\mathbb{F}|^{\mathbf{2(L-|S|)}+\delta}} \ll \frac{2^{c|\vec{m}_L|}}{|\mathbb{F}|^{\delta}}$$

Question: Fix $S \subseteq [c2^i]$, how many codewords are all zeros on S?

Minimum Distance Analysis

Result:
δ can be as small as $O(\frac{|\vec{m}_L|}{\log(|\mathbb{F}|)})$

Question: Fix $S \subseteq [c2^i]$, how many codewords are all zeros on S?

Case $|S| \geq L$: Only $\mathrm{Enc}_i(\vec{0})$ are all zeros on S
Case $|S| < L$: By **Rank-Nullity Thm**, there are at most $\approx |\mathbb{F}|^{(L-|S|)}$ possible $\mathrm{Enc}_i(\vec{x})$

$\vec{m}$ —Matrix map G→ $\mathrm{Enc}_i(\vec{m})[S]$ $\quad$ |Kernel(G)| ≤ ? $|\mathbb{F}|^{(L-|S|)}$

Rank-Nullity Thm

$\vec{m}$ —Matrix map $G' \supseteq G$→ $\mathrm{Enc}_i(\vec{m})[S']$ $\quad$ Kernel(G') = $\{0\}$

$|S'| = L$

Agenda

- IOPP from “foldable” linear codes
- Foldable linear codes over fast fields
- Efficient multilinear PCS compiler from BaseFold IOPPs

Multilinear PCS [KZG10,PST13]

pp ← Setup(d)

pp, $f \in \mathbb{F}_p^{(\leq 1)}[X_1, \ldots, X_d]$, r

Prover

commit(*pp*, f, r) ⇾ *"short"* C_f

pp

Verifier

Binding: cannot output two valid openings $(f_1, r_1), (f_2, r_2)$ for C_f.

Multilinear PCS [KZG10,PST13]

Evaluation phase | Goal: "convince" V that $f(\vec{z}) = y$ and $\boldsymbol{C_f}$ =<u>commit</u>(pp, f, r)

$\boldsymbol{x} = (pp, \boldsymbol{C_f}, \vec{z}, y)$, $\boldsymbol{w} = (\boldsymbol{f}, \boldsymbol{r})$

Prover

proof π

$\boldsymbol{x} = (pp, \boldsymbol{C_f}, \vec{z}, y)$

Verifier

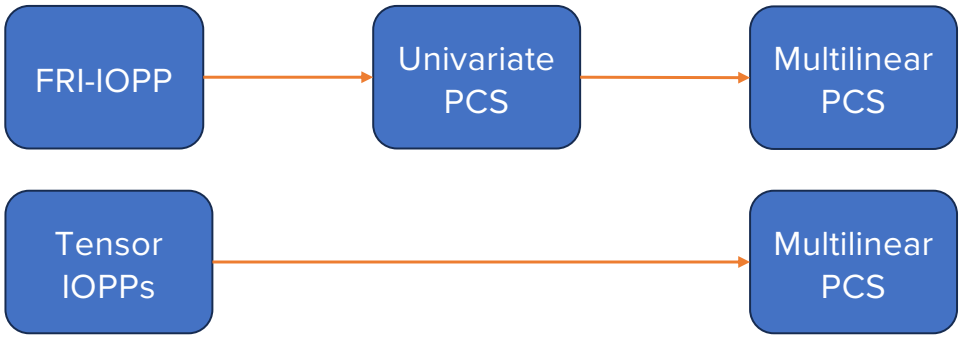
Accept/Reject

Eval is **knowledge sound**:

- V accepts ⇒ P "knows" $\boldsymbol{f} \in \mathbb{F}_p^{(\leq 1)}[X_1, \ldots, X_d], \boldsymbol{r}$ s.t. $f(\vec{z}) = y$ and $\boldsymbol{C_f}$ =<u>commit</u>(pp, f, r)

Multilinear PCS from BaseFold IOPPs

State-of-the-art:



Examples:
Gemini, HyperPlonk, ZeroMorph, etc

- fast V + "small" π :)
- not $\mathbb{F}$-agnostic : (

Examples:
Brakedown, Ligero, [Bootle et. al. 17], etc

- fast P + $\mathbb{F}$-agnostic :)
- slow V + large π : (

Our construction:

Advantages:
- Field agnostic!
- 3x prover improvement over the uni-to-multi PCS compilation
- Logarithmic verification/proof size

PCS Construction

Commitment to $f(X_1, \dots, X_d)$:
The Merkle commitment C to $\mathrm{Enc}(\vec{f})$

$\vec{f}$: coefficients of $f(X_1, \dots, X_d)$

Evaluation proof:

Goal: Given $\vec{z}, s$, "convince" V that $C = \mathrm{com}(f)$ and $f(\vec{z}) = s$ for some f

Multilinear Extension

Goal: "convince" V that $C = \mathrm{com}(f)$ and $f(\vec{z}) = \sum_{x \in B_d} f_z(\vec{x}) = s$

Sum-Check claim

$$f_z(\vec{x}) := f(\vec{x}) \cdot eq_{\vec{z}}(\vec{x})$$

$$eq_{\vec{z}}(\vec{x}) := \prod_{i=1}^{d} [z_i x_i + (1 - z_i x_i)]$$

Sumcheck for Polynomials [LFKN92]

V only makes 1 query to oracle f

Goal: "convince" V that $\sum_{x \in B_d} f(\vec{x}) = s$

check $f(r_1, \ldots, r_d) = h_1(r_1)$

Prover		Verifier
	$h_d(X) = \sum_{b \in B_{d-1}} f(b, X)$ →	check $h_d(0) + h_d(1) = s$
reduced claim: $\sum_{x \in B_{d-1}} f(\vec{x}, r_d) = h_d(r_d)$	← $r_d \leftarrow \mathbb{F}$	
	$h_{d-1}(X) = \sum_{b \in B_{d-2}} f(b, X, r_d)$ →	check $h_{d-1}(0) + h_{d-1}(1) = h_d(r_d)$
	← $r_{d-1} \leftarrow \mathbb{F}$	
	.....	
	.....	
	h_1 →	check $h_1(0) + h_1(1) = h_2(r_2)$

PCS Evaluation Proof

Goal: "convince" V that $C = \mathrm{com}(f)$ and $\sum_{x \in B_d} f_z(\vec{x}) = s$

$f_z(\vec{x}) \coloneqq f(\vec{x}) \cdot eq_{\vec{z}}(\vec{x})$

$\vec{f}$: coefficients of $f(X_1, \ldots, X_d)$

Committed Oracle: $C = \mathrm{Enc}(\vec{f})$

Prover

Verifier

reduced claim:
$\sum_{x \in B_{d-1}} f_z(\vec{x}, r_d) = h_d(r_d)$

$\vec{f}^{(d-1)} = \vec{f}_L + r_d \vec{f}_R$ is coeffs of $f(X_1, \ldots, X_{d-1}, r_d)$

Evaluation binding and **knowledge soundness** are nontrivial to prove

$h_d(X)$ →

check $h_d(0) + h_d(1) = s$

← $r_d \leftarrow \mathbb{F}$

check $f(r_1, \ldots, r_d) = h_1(r_1)$

Another PCS Eval query
Do we make any progress?

Oracle: $\pi^{(d-1)} = \mathrm{Enc}_{d-1}(\vec{f}^{(d-1)})$

$h_{d-1}(X)$ →

Reduced goal: "convince" V that $C_{(d-1)} = \mathrm{com}(f^{(d-1)})$ and $\sum_{x \in B_{d-1}} f_z^{(d-1)}(\vec{x}) = h_d(r_d)$

← $r_{d-1} \leftarrow \mathbb{F}$

.....

Oracle: $\pi^{(1)} = \mathrm{Enc}_1(\vec{f}^{(1)})$

$h_1(X)$ →

Consistency chk:

← $r_1 \leftarrow \mathbb{F}$

Run IOPP. Verify$^{C, \pi^{(d-1)}, \ldots, \pi^{(1)}}$

Oracle: $\pi^{(0)} = \mathrm{Enc}_0(\vec{f}^{(0)})$ →

$f_z(r_1, \ldots, r_d) = \vec{f}^{(0)} \cdot eq_{\vec{z}}(\vec{r}) = h_1(r_1)$

Evaluation Binding

Goal: "convince" V that $\exists f$: $C \approx \mathrm{com}(f)$ and $f(\vec{z}) = \sum_{x \in B_d} f(\vec{x}) eq_z(\vec{x}) = s$

Step 1: **IOPP soundness** → C is "close" to a codeword, i.e., $C \approx \mathrm{Enc}(\vec{f})$ for ***some unknown*** f

Step 2: **Sumcheck soundness** → $\sum_{x \in B_d} f(\vec{x}) eq_z(\vec{x}) = s$? **Hold on!**

Only if $\pi^{(0)} \approx \mathrm{Enc}_0(f(\vec{r}))$

Q1: How do we know that $\pi^{(0)} \approx \mathrm{Enc}_0(f(\vec{r}))$?

- Round oracles $\{\pi^{(i)}\}_{i=0}^{d}$ are all "close" to codewords w.h.p. $f^{(k+1)} := f(X_1, \dots, X_{k+1}, r_{k+2}, \dots, r_d)$
- Suppose not, then $\exists k: \pi^{(k+1)} \approx_\delta \mathrm{Enc}(f^{(k+1)})$ while $\pi^{(k)} \approx_\delta \mathrm{Enc}(g^{(k)})$
- By consistency chk, $\pi^{(k)}$ can't be too far from $\mathrm{fold}(\pi^{(k+1)}) \approx_\delta \mathrm{Enc}(f^{(k)})$

Two codewords are far from each other

Contradiction!

($\delta <$ unique decoding radius)

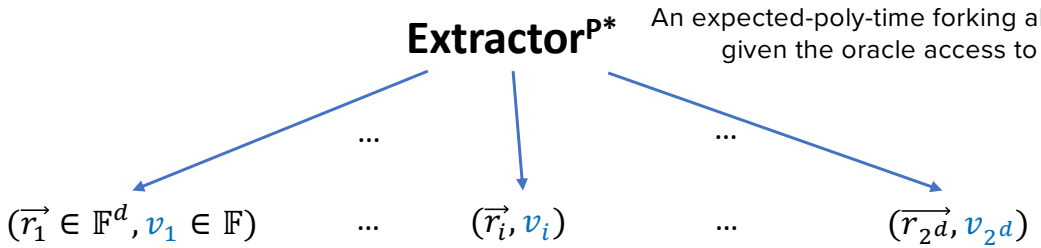
Q2: How to extract the knowledge of f in **poly-time**?

- Nontrivial as our code is not efficiently error-correct decodable : (

Knowledge Soundness

Goal: **Find** polynomial f: $C \approx \mathrm{com}(f)$ and $f(\vec{z}) = \sum_{x \in B_d} f(\vec{x}) eq_z(\vec{x}) = s$

Given **P*** that succeeds w/ a noticeable probability, construct



By a predicate forking lemma, with overwhelming probability:

- For all $i \in [2^d]$, $v_i = f(\vec{r_i}) = \langle \vec{f}, \mathrm{coef}(\vec{r_i}) \rangle$ (by evaluation binding property)
- $\left\{ \mathrm{coef}(\vec{r_i}) := \left(\prod_{j=1..d} \vec{r_i}[j]^{b_j} \right)_{b \in \{0,1\}^d} \in \mathbb{F}^{2^d} \right\}_{i=1}^{2^d}$ are linearly independent

A system of 2^d indep linear equations

Can recover $\vec{f}$ by Gaussian Elimination!

Summary & Future Work

Summary

- Generalize FRI IOPP to work for any “foldable” linear codes
- Fast & field-agnostic foldable linear codes
- Fast multilinear PCS from BaseFold IOPP for foldable codes

Open questions

- Linear-time encodable foldable codes
- Field-agnostic Maximum Distance Separable (MDS) foldable codes
- Generalize BaseFold IOPP to broader classes of codes (e.g., [BLNR22, ABN23])
- Combine with the “Binius” [DP23] trick for polynomials with small coefficients
- Better PCS soundness proof