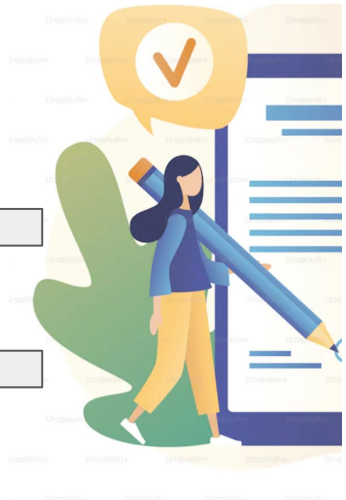
Basefold: Efficient Polynomial Commitment Schemes from Foldable Codes

Hadas Zeilberger, Binyi Chen, Ben Fisch

Polynomial Commitment Schemes

Prover

Verifier



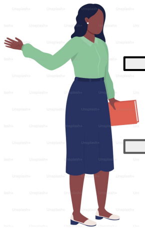
Com$(P(X_1,..,X_n))$

Evaluation point **v**

$P(v), \pi$

$b \in \{0,1\}$

Polynomial Commitment Schemes

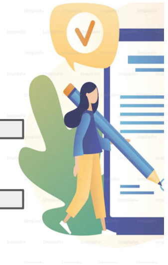


Com(P(X_1,..,X_n))

P(v), \pi

Evaluation point **v**

b \in {0,1}



<u>Binding Commitment</u>
With overwhelming probability, a commitment opens to at most one polynomial

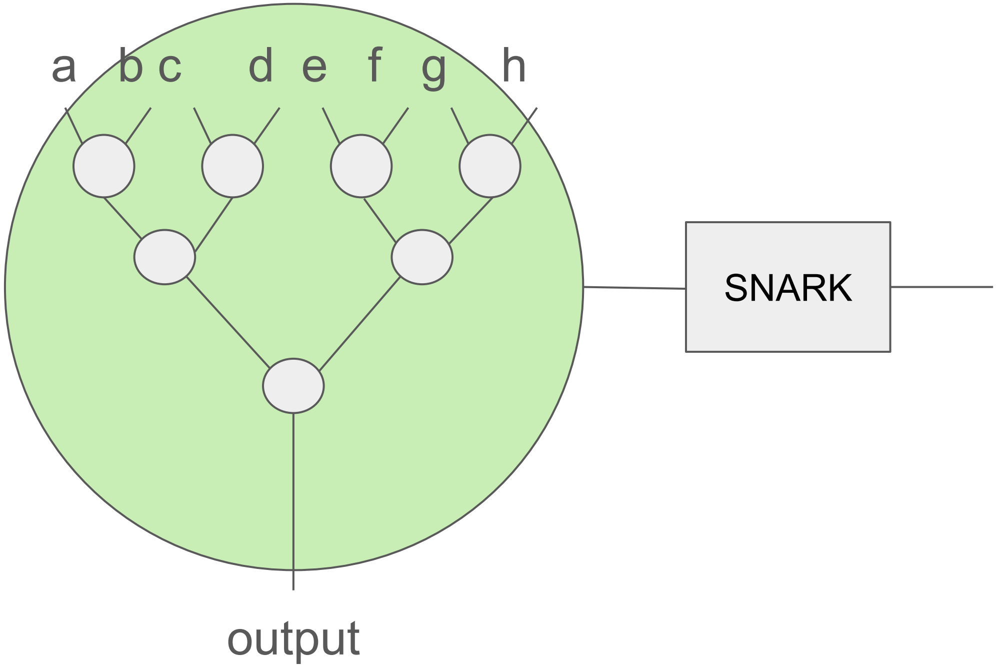
<u>Knowledge Sound Proof</u>
If a proof is "good quality" then there exists a polynomial time algorithm that can *extract* the correct polynomial from the proof

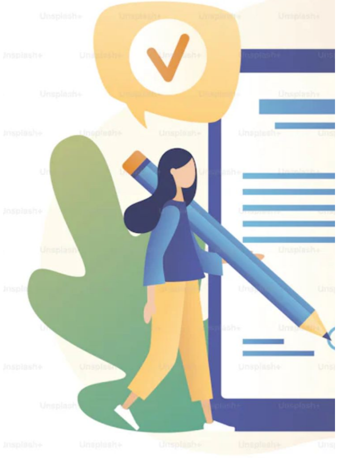
Polynomial Commitment Schemes

Applications

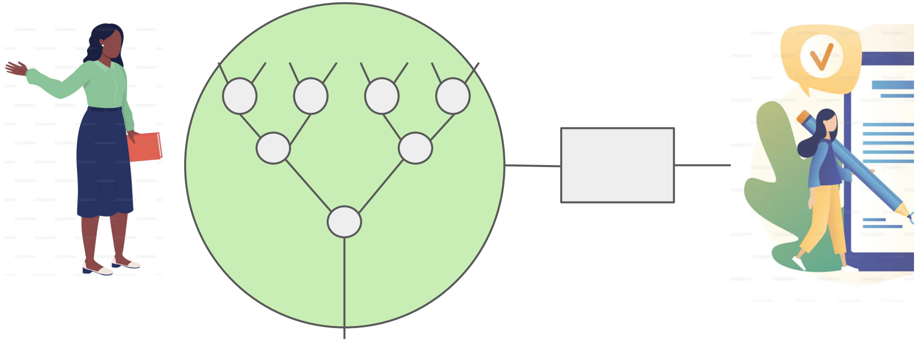
- Verifiable Secret Sharing
- Proof-of-storage
- **SNARKs**

Succinct **N**on-Interactive **Ar**gument of **K**nowledge





Succinct **N**on-Interactive **Ar**gument of **K**nowledge



<u>Security Properties:</u>

- Completeness
- Knowledge Soundness

<u>Efficiency Properties:</u>

- Succinctness
- Sublinear Time Verifier
- Small proofs

Problems with Existing SNARKs

Solutions with Efficient Verifiers

- Elliptic Curve Based: Require expensive multi-scalar multiplications over elliptic curves
- Code-Based: Require FFT-Friendly fields, which have a very specific algebraic structure, which introduces significant overhead for many important applications

Other Solutions

- MPC-Based (e.g. ZkBoo, Ligero) Fast prover, but have large proofs)
- Code-Based: e.g. Brakedown, Fast prover but requires proof size and verifier time that is O(sqrt(n))

Can We Do Better?

Multivariate PIOP (DARK, Hyperplonk, Spartan)

↓

Multilinear Polynomial Commitment Scheme

Low overhead over PCS, adopts field-choice of PCS
- Faster PCS -> Faster Prover

- **Efficient verifier,**
- **Efficient prover,**
- **Flexible over choice of field,**
- **Polylogarithmic proof size**

Can We Obtain A More Efficient *multilinear* Polynomial Commitment Scheme?

- Flexible field choices
- Polylog verifier
- Fast as possible prover

Our Solution: Basefold

- Efficient Multilinear PCS from Sumcheck and Generalized FRI
- New efficiently encododable code over *any finite field*:
    - Elliptic Curve Operations Using Only Native Field Operations (i.e. never have to do modulo operator in the arithmetic circuit)
    - Mersenne Primes*

Our Solution: Basefold

- **3x faster** than existing multilinear FRI constructions while maintaining polylogarithmic communication complexity
- **>20x faster** to prove signature verification circuits with no sacrifice in verifier costs
- In general, encoding elliptic curve operations into an arithmetic circuit using our code should be similarly efficient

Foldable Codes

Preliminaries: Error Correcting Codes

An [n,k] linear error-correcting code is equal to {v * G : v \in F^k} and G is an k x n matrix, called a *generator matrix.*

Example: A Reed-Solomon code, which are evaluations of a univariate polynomial over a domain D have generator matrix equal to the Vanadermonde matrix

Example: A [4,1] repetition code, which maps a -> a a a a, has generator matrix equal to [1, 1, 1, 1]

Our Solution: Basefold

Foldable Code

$$\begin{bmatrix} G_{i-1} & G_{i-1} \\ G_{i-1} * \mathbf{T1} & G_{i-1} * \mathbf{T2} \end{bmatrix}$$

$$\begin{bmatrix} t1[0] & & & & & \\ & \cdots & & & & \\ & & \cdots & & & \\ & & & \cdots & & \\ & & & & \cdots & \\ & & & t[2^{i-1}] & & \end{bmatrix}$$

Our Solution: Basefold

$$\begin{bmatrix} G_{i-1} & G_{i-1} \\ G_{i-1}*\mathbf{T1} & G_{i-1}*\mathbf{T2} \end{bmatrix}$$

<u>**Foldable Code**</u>

1) We can construct an **efficient multilinear PCS** from any foldable code

2) If we sample the T1,T2 randomly, then we obtain **a new *field-agnostic* new linear error-correcting code** with good distance properties

Technical Roadmap

- Definition of Foldable Code
- Multilinear PCS
  - Construction
  - High-level sketch of proof of knowledge soundness
- New Error-Correcting Code:
  - Construction
  - Distance Proof
- Applications and Benchmarks
- Future Directions and Open Problems

Foldable Codes

Preliminaries: Error Correcting Codes

An [n,k] linear error-correcting code is equal to {v * G : v \in F^k} and G is an k x n matrix, called a *generator matrix.*

Example: A Reed-Solomon code, which are evaluations of a univariate polynomial over a domain D have generator matrix equal to the Vanadermonde matrix

Example: A [4,1] repetition code, which maps a -> a a a a, has generator matrix equal to [1, 1, 1, 1]

Foldable Codes

<u>Recursive Generator Matrix</u>

$G_0 = [1 \, , \, 1]$

$$G_i := \begin{bmatrix} G_{i-1} & G_{i-1} \\ G_{i-1} * \mathbf{T1_i} & G_{i-1} * \mathbf{T2_i} \end{bmatrix}$$

$T1_i$, $T2_i$ are diagonal matrices for all i

$$G_0,..,G_i \longleftrightarrow (T1_1,T2_1),..,(T1_i,T2_i)$$

Foldable Codes

Recursive Encoding Algorithm

$G_0 = [1 , 1]$

$$G_i := \begin{bmatrix} G_{i-1} & G_{i-1} \\ G_{i-1}*\mathbf{T1}_i & G_{i-1}*\mathbf{T2}_i \end{bmatrix}$$

Diagonal Matrices
:$(T1_0,T2_0),..,(T1_i,T2_i)$

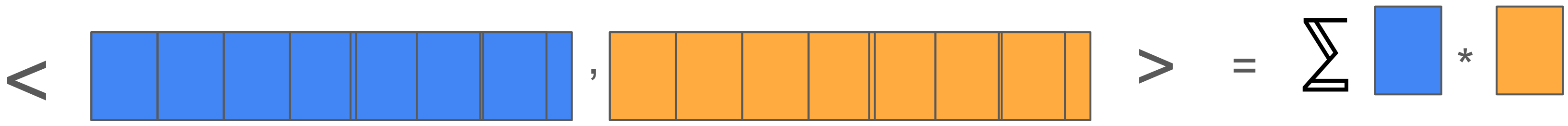
Enc(v) = v*G

Enc(v)[j] = <v,G[][j]>

Foldable Codes

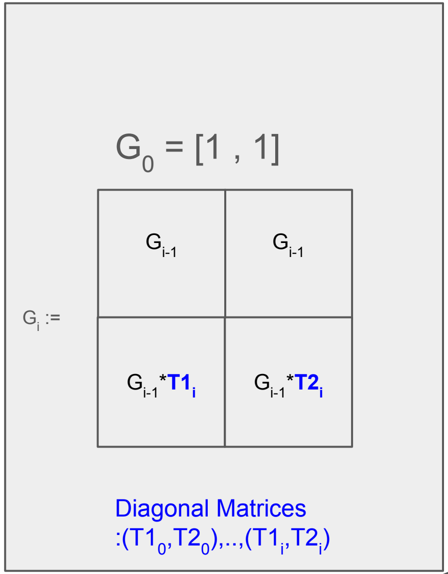
Recursive Encoding Algorithm

Inner Product Recap



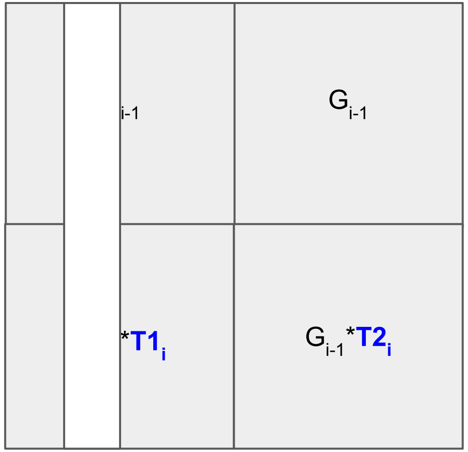
Foldable Codes

Recursive Encoding Algorithm



1,...,j,....n 1,...,j,....n

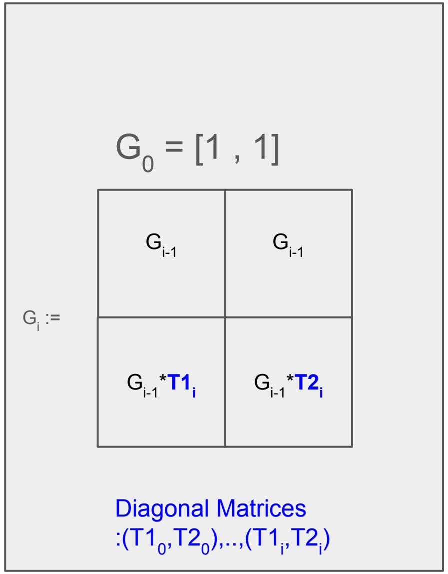
(v_L , v_R) *

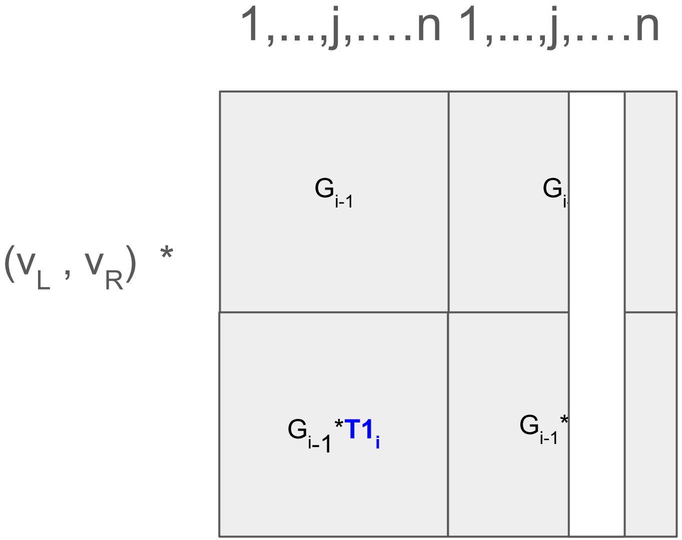


$<v_L,CJ> + t1[j]<v_R,CJ>$

Foldable Codes

Recursive Encoding Algorithm



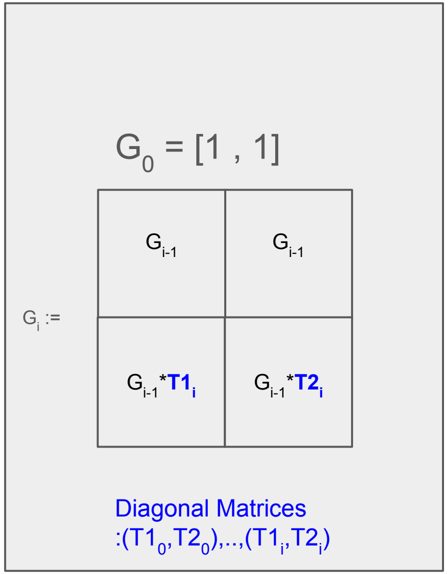


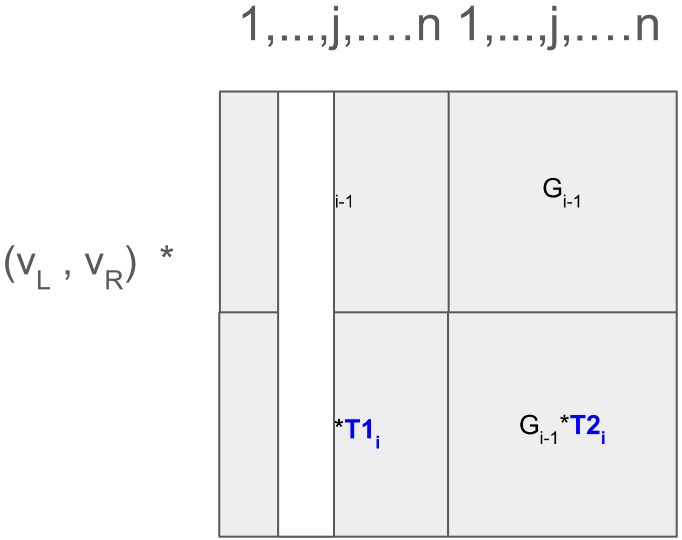
$\langle v_L, CJ\rangle + \mathbf{T1[j]}\langle v_R, CJ\rangle$

$\langle v_L, CJ\rangle + \mathbf{T2[j]}\langle v_R, CJ\rangle$

Foldable Codes

Recursive Encoding Algorithm





Let c = Enc_i (v_L,v_R)

c[j]:=Enc_{i-1}(v_L)[j] + **T1[j]**Enc_{i-1}(v_R)

c[j+n]:=Enc_{i-1}(v_L) + **T2[j]**Enc_{i-1}(v_R)

Foldable Codes

Multilinear Polynomial Evaluation

$G_0 = [1 , 1]$

$$G_i := \begin{bmatrix} G_{i-1} & G_{i-1} \\ G_{i-1}*\mathbf{T1}_i & G_{i-1}*\mathbf{T2}_i \end{bmatrix}$$

Diagonal Matrices :$(T1_0,T2_0),..,(T1_i,T2_i)$

$\mathbf{c}[i]:=Enc_{i-1}(v_L) + \mathbf{t1[j]}Enc_{i-1}(v_R)$

$\mathbf{c}[i]:=Enc_{i-1}(v_L) + \mathbf{t2[j]}Enc_{i-1}(v_R)$

Suppose $Enc_{i-1}(v_L) = P_L(\mathbf{x})$,

$Enc_{i-1}(v_R) = P_R(\mathbf{x})$

$\mathbf{c}[j] := P_L(x) + t1[j] * P_R(x) \mathbf{= P(x_1,..,x_{i-1},t1[j])}$

P(X_1,..,X_i) $= P_L(X_1,..,X_{i-1})+X_i*P_R(X_1,..,X_{i-1})$

Foldable Codes

Multilinear Polynomial Evaluation

Let $v_1,..,v_n$ \in F^k be the evaluation domain of C_{i-1}.

Then the evaluation domain of C_i is:
$(v_0 \| T1_i[0]),\ldots,(v\_n \| T1_i[n])$, $(v_0 \| T2_i[0],..,v_n \| T2_i[n])$

Foldable Codes

Multilinear Polynomial Evaluation

x_1
x_2
x_3
x_4

x_1, y_1
x_1, y_2

x_2, y_3
x_2, y_4

....

x_1, y_1, z_1
x_1, y_1, z_2

x_1, y_2, z_3
x_1, y_2, z_4

x_2, y_3, z_5
x_2, y_3, z_6

x_2, y_4, z_7
x_2, y_4, z_8

Foldable Codes

Multilinear Polynomial Evaluation

$P_{LL}(x_1), P_{LR}(x_1),$
$P_{RL}(x_1), P_{RR}(x_1)$

$P_L(x_1 , y_1), P_R(x_1 , y_1)$

$P_L(x_1 , y_1) + \mathbf{z_1} * P_R(x_1 , y_1)$

$P_L(x_1 , y_1) + \mathbf{z_2} * P_R(x_1 , y_1)$

$P_L(x_1 , y_2), P_R(x_1 , y_2)$

$P_L(x_1 , y_2) + \mathbf{z_3} * P_R(x_1 , y_2)$

$P_L(x_1 , y_2) + \mathbf{z_4} * P_R(x_1 , y_2)$

Polynomial Commitment Scheme from Foldable Codes

$P(X\_1, v_2, v_3)$

$P(X\_1, X\_2, v_3)$

$P(X\_1, X\_2, X\_3)$

$P(v_1, v_2, v_3)$

$P(x, v_2, v_3)$

$P(y, v_2, v_3)$

$P(x_1, y_1, v_3)$

$P(x_1 , y_2 , v_3)$

$P_L(x_1 , y_1) + \mathbf{z_1} * P_R(x_1 , y_1)$

$P_L(x_1 , y_1) + \mathbf{z_2} * P_R(x_1 , y_1)$

$P_L(x_1 , y_2) + \mathbf{z_3} * P_R(x_1 , y_2)$

$P_L(x_1 , y_2) + \mathbf{z_4} * P_R(x_1 , y_2)$

PCS from Foldable Codes

Building Block: IOP for Random Evaluation Point

P

V

$P_L(x_1, y_1) + \mathbf{z_1} * P_R(x_1, y_1)$

$P_L(x_1, y_1) + \mathbf{z_2} * P_R(x_1, y_1)$

$P_L(x_1, y_2) + \mathbf{z_3} * P_R(x_1, y_2)$

$P_L(x_1, y_2) + \mathbf{z_4} * P_R(x_1, y_2)$

$\mathbf{r_1}$

$P_L(x_1, y_1) + \mathbf{r_1} * P_R(x_1, y_1)$

$P_L(x_1, y_2) + \mathbf{r_1} * P_R(x_1, y_2)$

$\mathbf{r_2}$

$P(x_1, r_2, r_1)$

$\mathbf{r_3}$

$\mathbf{P(r_3, r_2, r_1)}$

PCS from Foldable Codes

Building Block: IOP for Random Evaluation Point

Key Point: Doubles as Proximity Test

Com

$P_L(x_1, y_1) + \mathbf{z_1} * P_R(x_1, y_1)$
$P_L(x_1, y_1) + \mathbf{z_2} * P_R(x_1, y_1)$
$P_L(x_1, y_2) + \mathbf{z_3} * P_R(x_1, y_2)$
$P_L(x_1, y_2) + \mathbf{z_4} * P_R(x_1, y_2)$

P

V

......

......

$\mathbf{P(r_3, r_2, r_1)}$

One field element repeated a constant number of times → **Com** is consistent with a polynomial in most places

PCS from Foldable Codes

Building Block: IOP for Random Evaluation Point

Key Point: Doubles as Proximity Test

Com

$P_L(x_1, y_1) + \mathbf{z_1} * P_R(x_1, y_1)$

$P_L(x_1, y_1) + \mathbf{z_2} * P_R(x_1, y_1)$

$P_L(x_1, y_2) + \mathbf{z_3} * P_R(x_1, y_2)$

Need to check within unique decoding radius each round

P .... **V**

$\mathbf{P(r_3, r_2, r_1)}$

One field element repeated a constant number of times → **Com** is consistent with a polynomial in most places

PCS from Foldable Codes

Building Block: IOP for Random Evaluation Point

Knowledge Soundness

$P(\mathbf{r}_1),\ldots,P(\mathbf{r}_n)$

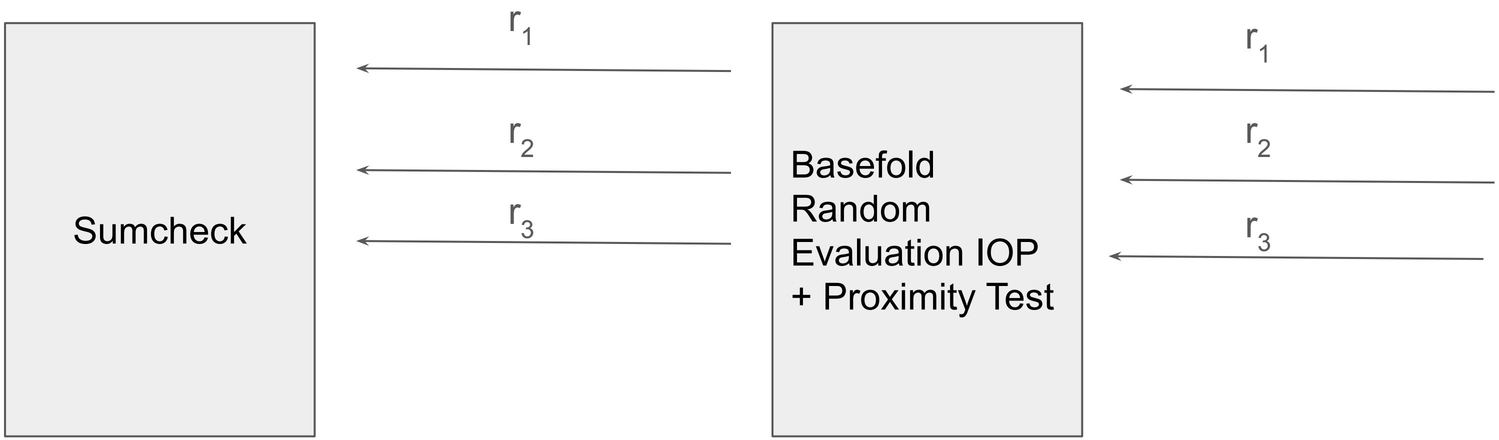
Interpolate P

$\mathbf{P(r_3, r_2, r_1)}$

PCS from Foldable Codes

IOP For *Any* Evaluation Point

Sumcheck for evaluation of Multilinear polynomial P reduces to checking random evaluation of P



PCS from Foldable Codes

Building Block: IOP for *any* Evaluation Point

Compile into SNARK via Fiat Shamir using Merkle Trees. In the end, we get:

- O(n) prover time (not including encoding time)
- O(log^2(n)) verifier time and proof size - (**~2x** bigger than FRI)
- **3x faster** than existing Multilinear FRI PCS

Open Problem: Prove that last oracle is an evaluation of a polynomial within list-decoding radius of the original oracle

Random Foldable Code

$G_0 = [1 , 1]$

$$G_i := \begin{bmatrix} G_{i-1} & G_{i-1} \\ G_{i-1} * \mathbf{T1_i} & G_{i-1} * \mathbf{T2_i} \end{bmatrix}$$

Diagonal Matrices
:$(T1_0,T2_0),..,(T1_i,T2_i)$

$$T1_i \xleftarrow{\$} F^{2\wedge i}$$

$$T2_i \xleftarrow{\$} F^{2\wedge i}$$

Random Foldable Code

Minimum Relative Distance of a random foldable code				
k_0	k_d	c	$\lvert\mathbb{F}\rvert$	Δ_{C_d}
2^5	2^{20}	16	2^{31}	.5044
1	2^{20}	16	2^{61}	.484
1	2^{25}	8	2^{128}	.557
1	2^{25}	8	2^{256}	.728

Reed-Solomon Code

c	Distance
2	0.5
4	~0.75
8	~0.875

Random Foldable Code

- Hamming distance is the number of *non zero entries* in a vector

  Example: [1,1,0,3] has hamming distance 3

  [1,0,0,3] has hamming distance 2

Random Foldable Code

- Multilinear Polynomials evaluated over random domains have good distance
- Schwartz-Zippel Lemma says P is 0 at a random point with probability d/|F|
- CDF of a binomial distribution: $F(r) = \sum_{x=Z}^{r} \binom{n}{x} p^x q^{(n-x)}$

$$F^{k} * \sum_{x=Z}^{r} \binom{n}{x} p^x q^{(n-x)}$$

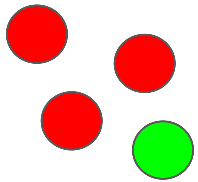
Random Foldable Code

- Multilinear Polynomials evaluated over random domains have good distance
- Schwartz-Zippel Lemma says P is 0 at a random point with probability d/|F|
- CDF of a binomial distribution: $F(r) = \sum_{x=Z}^{r} \binom{n}{x} p^x q^{(n-x)}$

Union Bound $\longrightarrow$ $\boxed{F^{k} *} \sum_{x=Z}^{r} \binom{n}{x} p^x q^{(n-x)}$

Random Foldable Code

Union Bound



The probability of sampling one green ball from two trials is smaller than equal to $(\frac{1}{4} + \frac{1}{4}) = \frac{1}{2}$

Random Foldable Code

$G_0 = [1 , 1]$

$$G_i := \begin{bmatrix} G_{i-1} & G_{i-1} \\ G_{i-1} * \mathbf{T1}_i & G_{i-1} * \mathbf{T2}_i \end{bmatrix}$$

Diagonal Matrices :$(T1_0,T2_0),..,(T1_i,T2_i)$

x_1
x_2
x_3
x_4

x_1 , y_1
x_1 , y_2

x_2 , y_3
x_2 , y_4

………

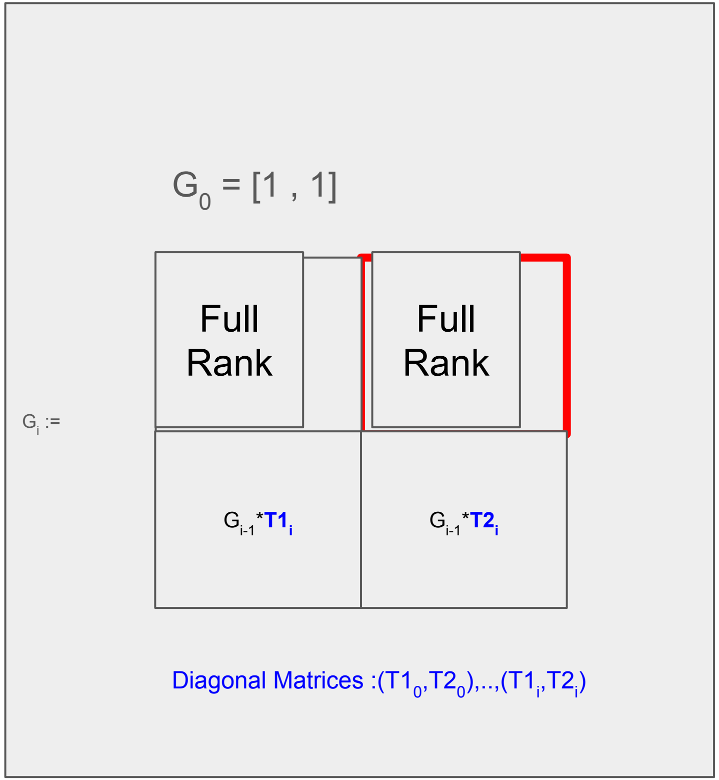
x_1 , y_1 , z_1
x_1 , y_1, z_2

x_1 , y_2, z_3
x_1 , y_2, z_4

x_2 , y_3, z_5
x_2 , y_3, z_6

x_2 , y_4, z_7
x_2 , y_4, z_8

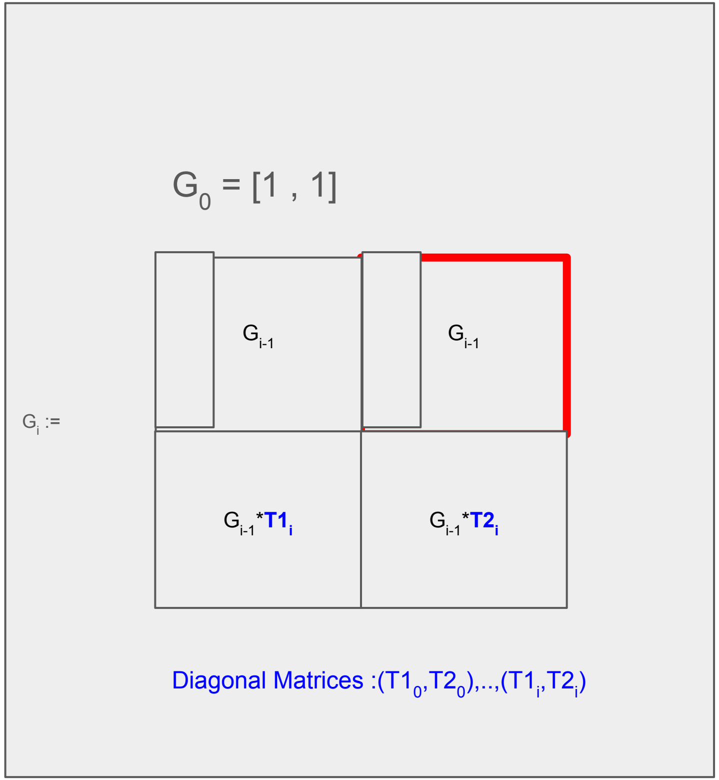
Random Foldable Code



- Let P be a polynomial that is 0 at every point associated with a column in the box
- Then for every column *not in the box*, P will be zero either on the left or the right with probability 1/F
-

$$F(r) = \sum_{x=Z}^{r} \binom{n}{x} p^x q^{(n-x)}$$

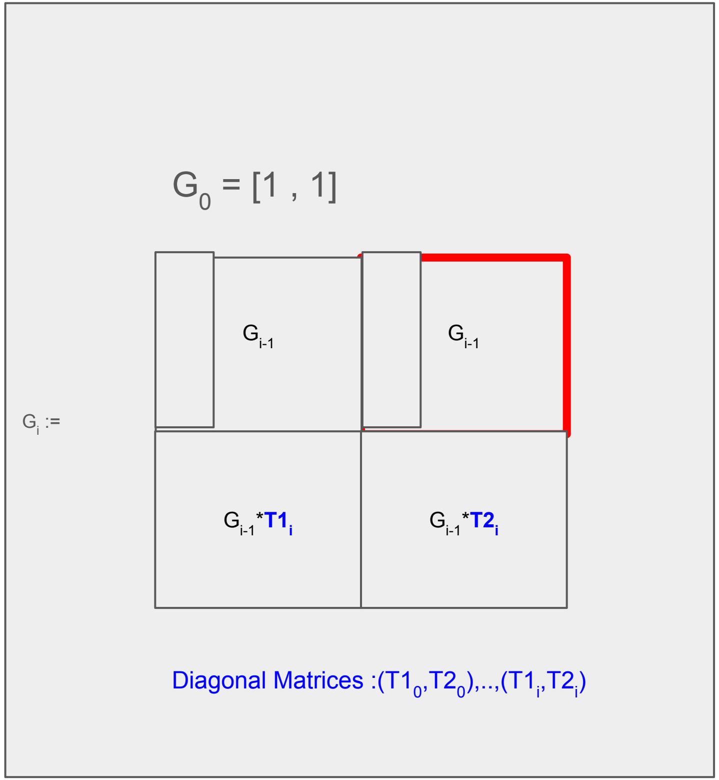
Random Foldable Code



- If the box is small, the probability is small, but there are more polynomials that are 0 on the entire box
- If the box is big, the probability is larger (Z is smaller), but there are *fewer* polynomials that are 0 on the entire box

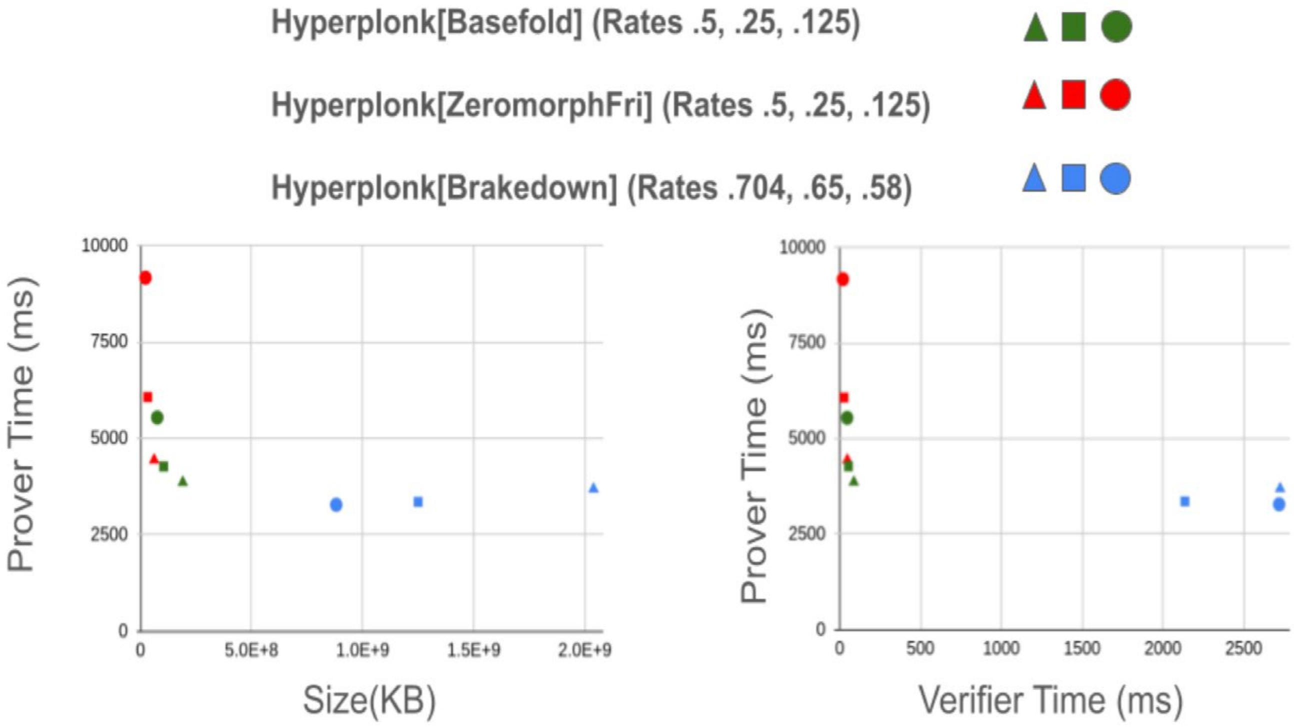
$$F(r) = \sum_{x=Z}^{r} \binom{n}{x} p^x q^{(n-x)}$$

Random Foldable Code



- If a Polynomial is 0 on a set of points in C_{i-1} domain, then those points are automatically 0 in the new domain
- Otherwise, we use CDF of the binomial distribution to bound the number of *new zeroes*
- We take a careful union bound, polynomials from larger sets have smaller probability of having >T zeroes

Benchmarks



Hyperplonk[Basefold] (Rates .5, .25, .125)
Hyperplonk[ZeromorphFri] (Rates .5, .25, .125)
Hyperplonk[Brakedown] (Rates .704, .65, .58)
Prover Time (ms)
10000
7500
5000
2500
0
0
5.0E+8
1.0E+9
1.5E+9
2.0E+9
Size(KB)
Prover Time (ms)
10000
7500
5000
2500
0
0
500
1000
1500
2000
2500
Verifier Time (ms)

Benchmarks

ECDSA Circuit			
Protocol	Prover Time (ms)	Proof Size (KB)	Verifier Time (ms)
Hyperplonk[Basefold]	122	6258	24
Hyperplonk[Brakedown]	168	32271	797
Hyperplonk[ZeromorphFri]	2888	7739	47
HyperPlonk[MKZG]	71027	7.74	107

Open Problems and Future Directions

Applications and Implementation

- Wrapping Basefold-based SNARK such as plonky3+Basefold into Groth16
    - Store Proof on chain
    - Endless recursion
- Use Interleaving trick directly with the SNARK

Protocol Improvements

- Use list-decoding regime instead of unique decoding regime
- Find better distance bounds or use distance boosting techniques on the code
- Find more efficient foldable codes