Basefold: Efficient Field-Agnostic Polynomial Commitment Schemes From Foldable Codes

Hadas Zeilberger

joint work with Binyi Chen(Stanford) and Ben Fisch(Yale)

Yale University

Basefold

- We formalize the definition of a *foldable code* and introduce a proof-of-proximity for any foldable code
- We construct a new family of linear codes, *random foldable codes*, and prove tight bounds on their minimum distance - **Field Agnosticity**
- We construct a new multilinear PCS by interleaving the proof-of-proximity with the classic sum-check protocol -> **New Efficient PCS**

SNARKs from Polynomial Interactive Oracle Proofs

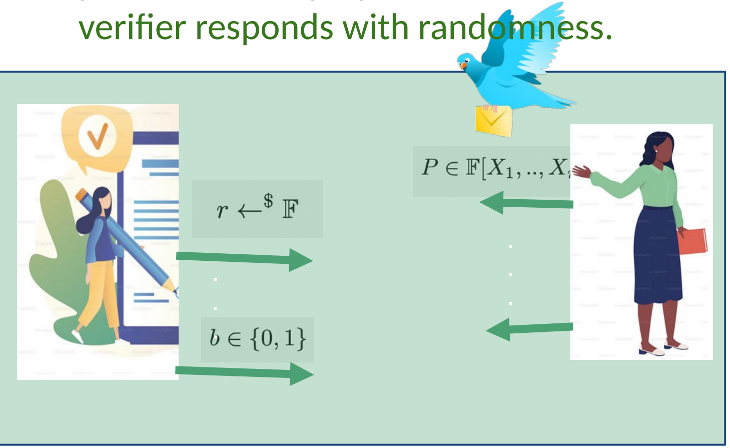
Multivariate PIOP
(Hyperplonk, Spartan, GKR)

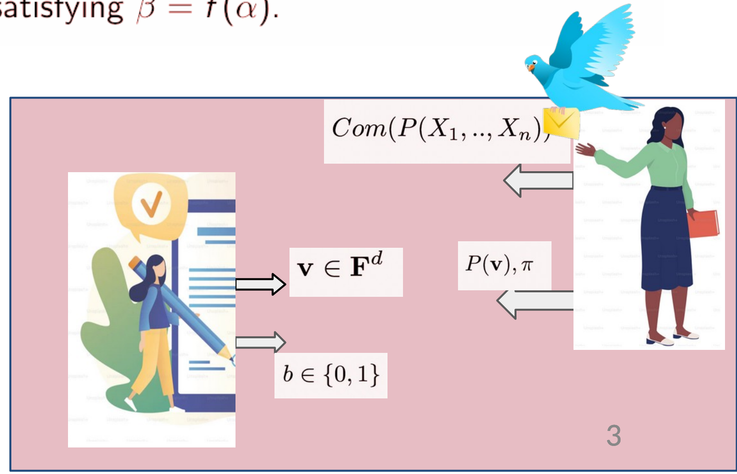
Multilinear PCS

SNARK

Polynomial Interactive Oracle Proof: A prover and a verifier interact over several rounds, in each round, the prover sends a polynomial, and the verifier responds with randomness.

Multilinear Polynomial Commitment Scheme: A prover commits to a polynomial $f \in \mathbb{F}[X_1, .., X_d]$ using a short commitment and later, given $\alpha, \beta \in \mathbb{F}$, sends a proof that it knows the d-variate multilinear polynomial satisfying $\beta = f(\alpha)$.





Hyperplonk and SuperSpartan (Multilinear PIOPs) [CBBZ23,GWC19]

- Multilinear PIOPs interpolate and evaluate polynomials over the *boolean hypercube*, rather than a subgroup, as in univariate PIOPs, (e.g. Plonk)
- Multilinear PIOPs replace computing a quotient polynomial - which requires an FFT with the more efficient sum-check protocol for multilinear evaluation
- **In Summary:** Multilinear PIOPs have lower prover overhead than univariate PIOPs, particularly for *high-degree gates*

Application	$\mathcal{R}_{R1CS}$	Ark-Spartan	$\mathcal{R}_{\text{PLONK+}}$	Jellyfish*	HyperPlonk
3-to-1 Rescue Hash	288 [1]	422 ms	144 [71]	40 ms	88 ms
PoK of Exponent	3315 [63]	902 ms	783 [63]	64 ms	105 ms
ZCash circuit	2^{17} [55]	8.3 s	2^{15} [42]	0.8 s	0.6 s
Zexe's recursive circuit	2^{22} [81]	6 min	2^{17} [81]	13.1 s	5.1 s
Rollup of 50 private tx	2^{25}	39 min[b]	2^{20} [71]	110 s	38.2 s
zkEVM circuit[a]	N/A	N/A	2^{27}	1 hour[b,c]	25 min[b,c]

*Jellyfish is an implementation of Plonk

Problems with Existing Multilinear PCS Constructions

Options for Multilinear PCS are limited.

- **High verifier costs**/requires proof recursion with high overhead
- Relies on **univariate-multilinear transformation**, which requires a constant number of univariate commitments
- **Limited field choices** forces many applications to use non-native field operations, which requires encoding the “modulo p” operation for *each multiplication* (leading to an overhead of up to 256 constraints)

Polynomial Commitment Schemes from Error Correcting Codes

- A linear $[n, k]$ error correcting code $(n, k \in \mathbb{F}^n, n > k)$ is a k dimensional subspace of $\mathbb{F}^n$
- The Hamming distance between two vectors, $\mathbf{v}, \mathbf{w}$ is

$$|\{i \in [1, n] : v[i] \neq w[i]\}$$

- A *proof of proximity* enables a prover to convince a verifier that is knows a vector that is "close" in Hamming distance to a codeword

FRI IOPP

[BBHR18]

- The FRI (Interactive Oracle) Proof of Proximity (IOPP) uses a "split and fold" approach
- In each round, the prover sends an oracle to a codeword that is half the size of the oracle from the previous round
- Finally, after $d = log(n)$ rounds, the verifier receives a constant-sized vector, which it can read in full

r_d

r_{d-1}

$\vdots$

r_0

$Enc(P)$

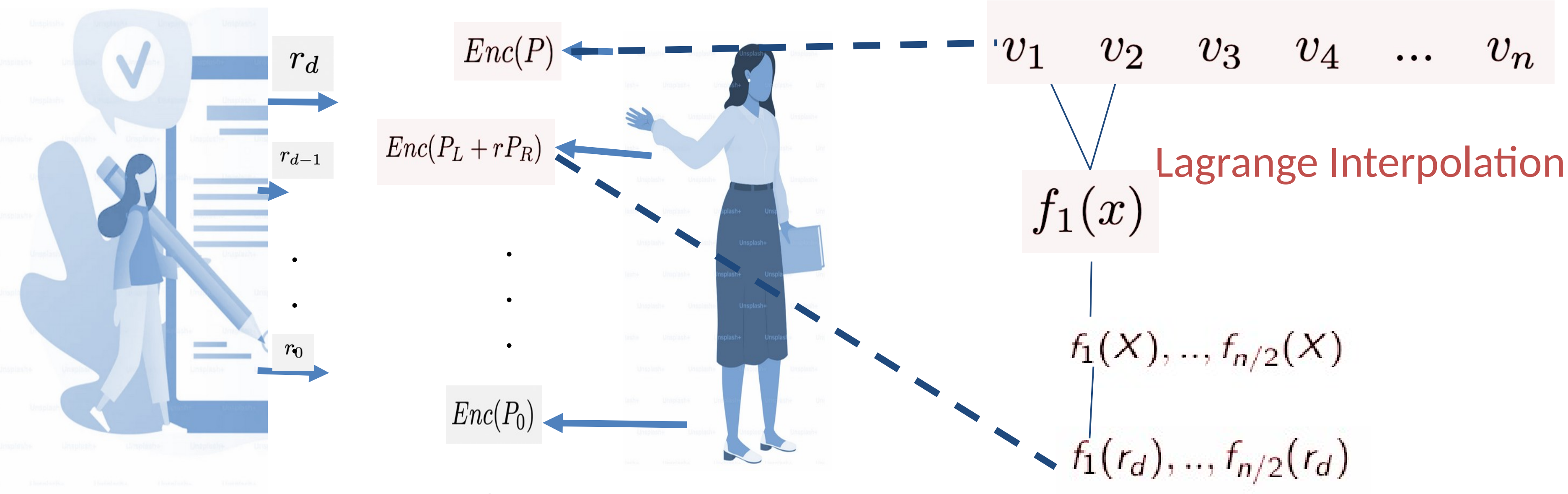
$Enc(P_L + rP_R)$

$\vdots$

$Enc(P_0)$

FRI IOPP

How to do this in linear time?



Technical Roadmap

- <u>Foldable *Codes*</u>
- Random Foldable Codes
- Multilinear PCS Construction from Foldable Codes

Foldable Codes

- Every linear code can be described in terms of a generator matrix, G such that $Enc(v) = v \cdot G$.
- Foldable codes have a generator matrix with the following structure:

$$\begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d \end{bmatrix}$$

$(T_d, T'_d), .., (T_0, T'_0)$ are diagonal matrices

Foldable Codes

Starting Point: Viewing RS Codes over FFT Friendly fields as a *Foldable Code*

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_0 & x_1 & x_2 & \dots & x_n \\ (x_0)^2 & (x_1)^2 & (x_2)^2 & \dots & (x_n)^2 \\ \dots \\ \dots \\ (x_0)^d & (x_1)^d & (x_2)^d & \dots & (x_n)^d \end{bmatrix} \longrightarrow \begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d \end{bmatrix}$$

Foldable Codes

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
g^0 & g^1 & g^2 & g^3 & g^4 & g^5 & g^6 & g^7 \\
(g^0)^2 & (g^1)^2 & (g^2)^2 & (g^3)^2 & (g^4)^2 & (g^5)^2 & (g^6)^2 & (g^7)^2 \\
(g^0)^3 & (g^1)^3 & (g^2)^3 & (g^3)^3 & (g^4)^3 & (g^5)^3 & (g^6)^3 & (g^7)^3 \\
(g^0)^4 & (g^1)^4 & (g^2)^4 & (g^3)^4 & (g^4)^4 & (g^5)^4 & (g^6)^4 & (g^7)^4 \\
(g^0)^5 & (g^1)^5 & (g^2)^5 & (g^3)^5 & (g^4)^5 & (g^5)^5 & (g^6)^5 & (g^7)^5 \\
(g^0)^6 & (g^1)^6 & (g^2)^6 & (g^3)^6 & (g^4)^6 & (g^5)^6 & (g^6)^6 & (g^7)^6 \\
(g^0)^7 & (g^1)^7 & (g^2)^7 & (g^3)^7 & (g^4)^7 & (g^5)^7 & (g^6)^7 & (g^7)^7
\end{bmatrix}
\longrightarrow
\begin{bmatrix}
g^0 & g^0 & g^0 & g^0 & g^0 & g^0 & g^0 & g^0 \\
g^0 & g^1 & g^2 & g^3 & g^4 & g^5 & g^6 & g^7 \\
g^0 & g^2 & g^4 & g^6 & g^0 & g^2 & g^4 & g^6 \\
g^0 & g^3 & g^6 & g^1 & g^4 & g^7 & g^2 & g^5 \\
g^0 & g^4 & g^0 & g^4 & g^0 & g^4 & g^0 & g^4 \\
g^0 & g^5 & g^2 & g^7 & g^4 & g^1 & g^6 & g^3 \\
g^0 & g^6 & g^4 & g^2 & g^0 & g^6 & g^4 & g^2 \\
g^0 & g^7 & g^6 & g^5 & g^4 & g^3 & g^2 & g^1
\end{bmatrix}
$$

$$
\begin{bmatrix}
G_{d-1} & G_{d-1} \\
G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d
\end{bmatrix}
$$

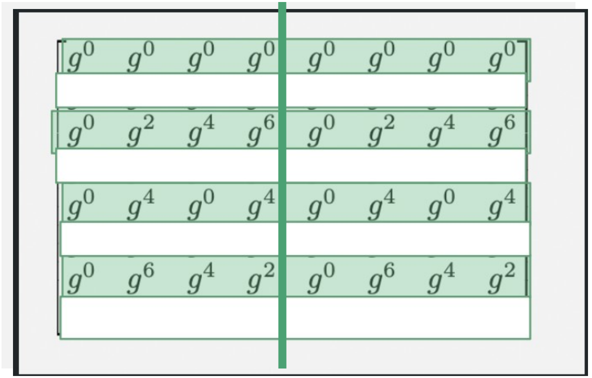
Foldable Codes

$$
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
g^0 & g^1 & g^2 & g^3 & g^4 & g^5 & g^6 & g^7 \\
(g^0)^2 & (g^1)^2 & (g^2)^2 & (g^3)^2 & (g^4)^2 & (g^5)^2 & (g^6)^2 & (g^7)^2 \\
(g^0)^3 & (g^1)^3 & (g^2)^3 & (g^3)^3 & (g^4)^3 & (g^5)^3 & (g^6)^3 & (g^7)^3 \\
(g^0)^4 & (g^1)^4 & (g^2)^4 & (g^3)^4 & (g^4)^4 & (g^5)^4 & (g^6)^4 & (g^7)^4 \\
(g^0)^5 & (g^1)^5 & (g^2)^5 & (g^3)^5 & (g^4)^5 & (g^5)^5 & (g^6)^5 & (g^7)^5 \\
(g^0)^6 & (g^1)^6 & (g^2)^6 & (g^3)^6 & (g^4)^6 & (g^5)^6 & (g^6)^6 & (g^7)^6 \\
(g^0)^7 & (g^1)^7 & (g^2)^7 & (g^3)^7 & (g^4)^7 & (g^5)^7 & (g^6)^7 & (g^7)^7
\end{bmatrix}
\longrightarrow
\begin{bmatrix}
g^0 & g^0 & g^0 & g^0 & g^0 & g^0 & g^0 & g^0 \\
 & & & & & & & \\
g^0 & g^2 & g^4 & g^6 & g^0 & g^2 & g^4 & g^6 \\
 & & & & & & & \\
g^0 & g^4 & g^0 & g^4 & g^0 & g^4 & g^0 & g^4 \\
 & & & & & & & \\
g^0 & g^6 & g^4 & g^2 & g^0 & g^6 & g^4 & g^2 \\
 & & & & & & &
\end{bmatrix}
$$

$$
\begin{bmatrix}
G_{d-1} & G_{d-1} \\
G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d
\end{bmatrix}
$$

Foldable Codes

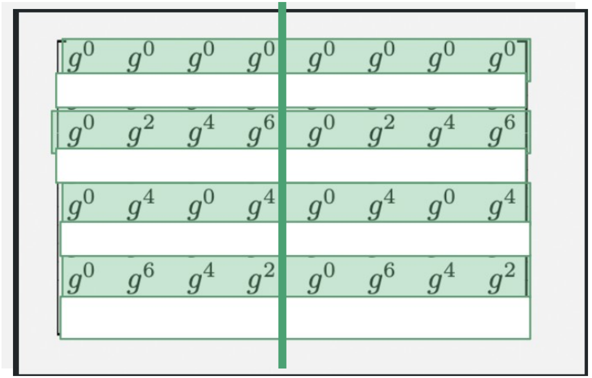
$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ g^0 & g^1 & g^2 & g^3 & g^4 & g^5 & g^6 & g^7 \\ (g^0)^2 & (g^1)^2 & (g^2)^2 & (g^3)^2 & (g^4)^2 & (g^5)^2 & (g^6)^2 & (g^7)^2 \\ (g^0)^3 & (g^1)^3 & (g^2)^3 & (g^3)^3 & (g^4)^3 & (g^5)^3 & (g^6)^3 & (g^7)^3 \\ (g^0)^4 & (g^1)^4 & (g^2)^4 & (g^3)^4 & (g^4)^4 & (g^5)^4 & (g^6)^4 & (g^7)^4 \\ (g^0)^5 & (g^1)^5 & (g^2)^5 & (g^3)^5 & (g^4)^5 & (g^5)^5 & (g^6)^5 & (g^7)^5 \\ (g^0)^6 & (g^1)^6 & (g^2)^6 & (g^3)^6 & (g^4)^6 & (g^5)^6 & (g^6)^6 & (g^7)^6 \\ (g^0)^7 & (g^1)^7 & (g^2)^7 & (g^3)^7 & (g^4)^7 & (g^5)^7 & (g^6)^7 & (g^7)^7 \end{bmatrix} \longrightarrow$$



$$\begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d \end{bmatrix}$$

Foldable Codes

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ g^0 & g^1 & g^2 & g^3 & g^4 & g^5 & g^6 & g^7 \\ (g^0)^2 & (g^1)^2 & (g^2)^2 & (g^3)^2 & (g^4)^2 & (g^5)^2 & (g^6)^2 & (g^7)^2 \\ (g^0)^3 & (g^1)^3 & (g^2)^3 & (g^3)^3 & (g^4)^3 & (g^5)^3 & (g^6)^3 & (g^7)^3 \\ (g^0)^4 & (g^1)^4 & (g^2)^4 & (g^3)^4 & (g^4)^4 & (g^5)^4 & (g^6)^4 & (g^7)^4 \\ (g^0)^5 & (g^1)^5 & (g^2)^5 & (g^3)^5 & (g^4)^5 & (g^5)^5 & (g^6)^5 & (g^7)^5 \\ (g^0)^6 & (g^1)^6 & (g^2)^6 & (g^3)^6 & (g^4)^6 & (g^5)^6 & (g^6)^6 & (g^7)^6 \\ (g^0)^7 & (g^1)^7 & (g^2)^7 & (g^3)^7 & (g^4)^7 & (g^5)^7 & (g^6)^7 & (g^7)^7 \end{bmatrix}$$



$$\begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d \end{bmatrix}$$

Even Rows $=$ $$\begin{bmatrix} g^0 & g^0 & g^0 & g^0 \\ g^0 & g^2 & g^4 & g^6 \\ g^0 & g^4 & g^0 & g^4 \\ g^0 & g^6 & g^4 & g^2 \end{bmatrix}$$

Foldable Codes

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ g^0 & g^1 & g^2 & g^3 & g^4 & g^5 & g^6 & g^7 \\ (g^0)^2 & (g^1)^2 & (g^2)^2 & (g^3)^2 & (g^4)^2 & (g^5)^2 & (g^6)^2 & (g^7)^2 \\ (g^0)^3 & (g^1)^3 & (g^2)^3 & (g^3)^3 & (g^4)^3 & (g^5)^3 & (g^6)^3 & (g^7)^3 \\ (g^0)^4 & (g^1)^4 & (g^2)^4 & (g^3)^4 & (g^4)^4 & (g^5)^4 & (g^6)^4 & (g^7)^4 \\ (g^0)^5 & (g^1)^5 & (g^2)^5 & (g^3)^5 & (g^4)^5 & (g^5)^5 & (g^6)^5 & (g^7)^5 \\ (g^0)^6 & (g^1)^6 & (g^2)^6 & (g^3)^6 & (g^4)^6 & (g^5)^6 & (g^6)^6 & (g^7)^6 \\ (g^0)^7 & (g^1)^7 & (g^2)^7 & (g^3)^7 & (g^4)^7 & (g^5)^7 & (g^6)^7 & (g^7)^7 \end{bmatrix} \longrightarrow \begin{bmatrix} g^0 & g^0 & g^0 & g^0 & g^0 & g^0 & g^0 & g^0 \\ g^0 \cdot g^0 & g^0 \cdot g^1 & g^0 \cdot g^2 & g^0 \cdot g^3 & g^0 \cdot g^4 & g^0 \cdot g^5 & g^0 \cdot g^6 & g^0 \cdot g^7 \\ g^0 & g^2 & g^4 & g^6 & g^0 & g^2 & g^4 & g^6 \\ g^0 \cdot g^0 & g^2 \cdot g^1 & g^4 \cdot g^2 & g^6 \cdot g^3 & g^0 \cdot g^4 & g^2 \cdot g^5 & g^4 \cdot g^6 & g^6 \cdot g^7 \\ g^0 & g^4 & g^0 & g^4 & g^0 & g^4 & g^0 & g^4 \\ g^0 \cdot g^0 & g^4 \cdot g^1 & g^0 \cdot g^2 & g^4 \cdot g^3 & g^0 \cdot g^4 & g^4 \cdot g^5 & g^0 \cdot g^6 & g^4 \cdot g^7 \\ g^0 & g^6 & g^4 & g^2 & g^0 & g^6 & g^4 & g^2 \\ g^0 \cdot g^0 & g^6 \cdot g^1 & g^4 \cdot g^2 & g^2 \cdot g^3 & g^0 \cdot g^4 & g^6 \cdot g^5 & g^4 \cdot g^6 & g^2 \cdot g^7 \end{bmatrix}$$

$$\text{Odd Rows} = \begin{bmatrix} g^0 & g^0 & g^0 & g^0 \\ g^0 & g^2 & g^4 & g^6 \\ g^0 & g^4 & g^0 & g^4 \\ g^0 & g^6 & g^4 & g^2 \end{bmatrix} \times \begin{bmatrix} g^0 & 0 & 0 & 0 \\ 0 & g^1 & 0 & 0 \\ 0 & 0 & g^2 & 0 \\ 0 & 0 & 0 & g^3 \end{bmatrix}$$

Foldable Codes

$$
\begin{bmatrix}
g^0 & g^0 & g^0 & g^0 & g^0 & g^0 & g^0 & g^0 \\
g^0 \cdot g^0 & g^0 \cdot g^1 & g^0 \cdot g^2 & g^0 \cdot g^3 & g^0 \cdot g^4 & g^0 \cdot g^5 & g^0 \cdot g^6 & g^0 \cdot g^7 \\
g^0 & g^2 & g^4 & g^6 & g^0 & g^2 & g^4 & g^6 \\
g^0 \cdot g^0 & g^2 \cdot g^1 & g^4 \cdot g^2 & g^6 \cdot g^3 & g^0 \cdot g^4 & g^2 \cdot g^5 & g^4 \cdot g^6 & g^6 \cdot g^7 \\
g^0 & g^4 & g^0 & g^4 & g^0 & g^4 & g^0 & g^4 \\
g^0 \cdot g^0 & g^4 \cdot g^1 & g^0 \cdot g^2 & g^4 \cdot g^3 & g^0 \cdot g^4 & g^4 \cdot g^5 & g^0 \cdot g^6 & g^4 \cdot g^7 \\
g^0 & g^6 & g^4 & g^2 & g^0 & g^6 & g^4 & g^2 \\
g^0 \cdot g^0 & g^6 \cdot g^1 & g^4 \cdot g^2 & g^2 \cdot g^3 & g^0 \cdot g^4 & g^6 \cdot g^5 & g^4 \cdot g^6 & g^2 \cdot g^7
\end{bmatrix}
\longrightarrow
\begin{bmatrix}
G_{d-1} & G_{d-1} \\
G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d
\end{bmatrix}
$$

Technical Roadmap

- Foldable *Codes*
- <u>Random Foldable Codes</u>
- Multilinear PCS Construction from Foldable Codes

Random Foldable Codes

A Random foldable code is a foldable code where the elements of the diagonal matrices are uniformly sampled from $\mathbb{F}$

$$\begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d \end{bmatrix}$$

Random Foldable Codes

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

Table 1: The relative minimum distances of random foldable codes.

Technical Roadmap

- Foldable *Codes*
- Random Foldable Codes
- Multilinear PCS Construction from Foldable Codes

Foldable Codes as Punctured Reed-Muller Codes

$$(m_l || m_r) \cdot \begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d \end{bmatrix}$$

$$\{m \cdot G_{d-1} : m \in \mathbb{F}^{2^{d-1}}\} = \{(P(\vec{v}_1), .., P(\vec{v}_n)) : P \in \mathbb{F}[X_1, .., X_{d-1}]\}$$

Foldable Codes as Punctured Reed-Muller Codes

$$\{m \cdot G_{d-1} : m \in \mathbb{F}^{2^{d-1}}\} = \{(P(\vec{v}_1), .., P(\vec{v}_n)) : P \in \mathbb{F}[X_1, .., X_{d-1}]\}$$

$$(m_l || m_r) \cdot \begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d \end{bmatrix}$$

$$\{P_{m_l}(\vec{v}_i) + T_{d-1}[i] P_{m_r}(\vec{v}_i) : i \in [1, n]\}$$

$$= \{P^*(\vec{v}_i, T_{d-1}[i]) : i \in [1, n]\}$$

Foldable Codes as Punctured Reed-Muller Codes

$$\begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T_d' \end{bmatrix}$$

$$\{P^*(\vec{v}_i, T_d[i], i \in [1, n/2]\}$$

$\vec{v}_1$	$\vec{v}_2$	$\vec{v}_3$	$\vec{v}_4$	$\vec{v}_5$	$\vec{v}_6$	$\vec{v}_7$	$\vec{v}_8$
x_1	x_2	x_1	x_2	x_1	x_2	x_1	x_2
y_1	y_2	y_3	y_4	y_1	y_2	y_3	y_4
z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8

Foldable Codes as Punctured Reed-Muller Codes

$$\begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d \end{bmatrix}$$

$$\{P^*(\vec{v}_i, T_d[i], i \in [1, n/2]\}$$

$\vec{v}_1$	$\vec{v}_2$	$\vec{v}_3$	$\vec{v}_4$	$\vec{v}_5$	$\vec{v}_6$	$\vec{v}_7$	$\vec{v}_8$
x_1	x_2	x_1	x_2	x_1	x_2	x_1	x_2
y_1	y_2	y_3	y_4	y_1	y_2	y_3	y_4
z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8

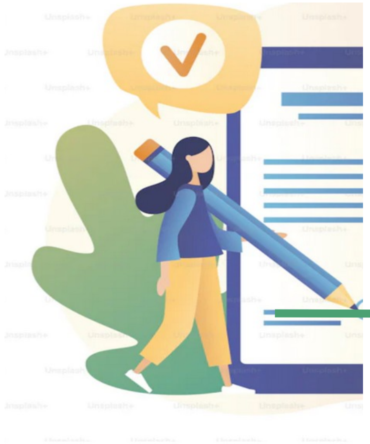
Foldable Codes as Punctured Reed-Muller Codes

$$\begin{bmatrix} G_{d-1} & G_{d-1} \\ G_{d-1} \cdot T_d & G_{d-1} \cdot T'_d \end{bmatrix}$$

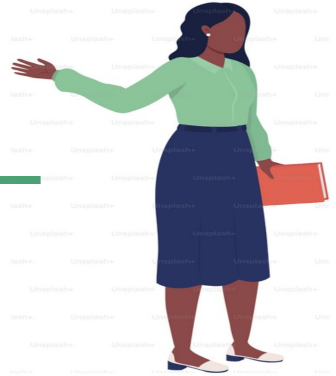
$$\{P^*(\vec{v}_i, T_d[i], i \in [1, n/2]\}$$

$\vec{v}_1$	$\vec{v}_2$	$\vec{v}_3$	$\vec{v}_4$	$\vec{v}_5$	$\vec{v}_6$	$\vec{v}_7$	$\vec{v}_8$
x_1	x_2	x_1	x_2	x_1	x_2	x_1	x_2
y_1	y_2	y_3	y_4	y_1	y_2	y_3	y_4
z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8

Foldable Codes -> Multilinear PCS



$$\{P(v_i) : i \in [1,8]\}$$



$$\vec{r}$$

$P(x_1, y_1, z_1)$ $P(x_1, y_1, z_5)$

$\vec{v}_1$	$\vec{v}_2$	$\vec{v}_3$	$\vec{v}_4$	$\vec{v}_5$	$\vec{v}_6$	$\vec{v}_7$	$\vec{v}_8$
x_1	x_2	x_1	x_2	x_1	x_2	x_1	x_2
y_1	y_2	y_3	y_4	y_1	y_2	y_3	y_4
z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8

Lagrange Interpolation

$$P(x, y, Z) \rightarrow P(x, y, \vec{r}[0])$$

$$= (P_L + rP_R)(x_1, y_1)$$

Foldable Codes -> Multilinear PCS

$$\{P(v_i) : i \in [1, 8]\}$$

$$P(r_1, r_2, r_3)$$

Foldable Codes -> Multilinear PCS

- Reduce evaluation check of a multilinear polynomial at a generic point to a evaluation check at a *random* point using sum-check protocol + multilinear extension

$$P(X_1, .., X_d) = \sum_{\mathbf{b} \in \{0,1\}^d} P(\mathbf{b}) \cdot \tilde{eq}_{\mathbf{b}}(X_1, .., X_d)$$

$$\Big\downarrow \text{ d rounds}$$

$$f(\vec{r})$$

Foldable Codes -> Multilinear PCS

Basefold IOPP and sum-check are then interleaved, sharing the same verifier randomness. The last oracle of the Basefold IOPP is the random query to the polynomial oracle.

$$P(X_1, .., X_d) = \sum_{\mathbf{b} \in \{0,1\}^d} P(\mathbf{b}) \cdot \tilde{eq}_{\mathbf{b}}(X_1, .., X_d)$$

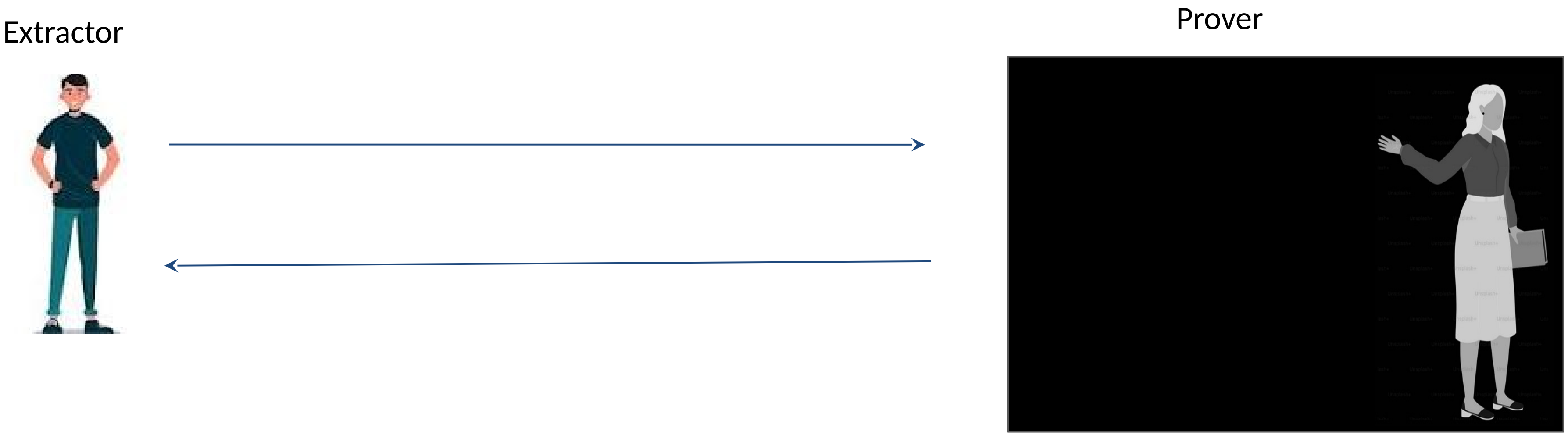
$$\{P(v_i) : i \in [1, 8]\}$$

$$P(r_1, r_2, r_3)$$

$$P(\mathbf{r})$$

Knowledge Soundness

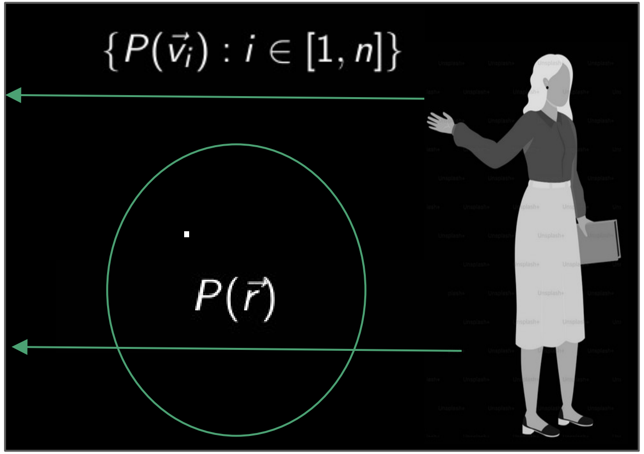
- **Lemma**: For any prover strategy that passes the verifier checks with non-negligible probability, there exists a polynomial time extractor with black-box access to the prover, that outputs the underlying polynomial



Foldable Codes -> Multilinear PCS

- **BasefoldPCS with a Reed-Solomon code:** the knowledge extractor queries sufficiently many locations from the Merkle tree and decodes
- **BasefoldPCS with Random Foldable Code:** extractor queries the prover for enough random evaluations of the polynomial to interpolate the coefficients of the underlying polynomial.

$P(r_i)$



$\{P(\vec{r}_j) : j \in [1, poly(k)]\}$

Foldable Codes -> Multilinear PCS

- Basefold IOPP needs to prove that *P(r)* is the evaluation of the polynomial underlying the prover's commitment
- To do this, the verifier needs to check each oracle within the *unique decoding radius*, rather than the *list-decoding radius*, as in FRI

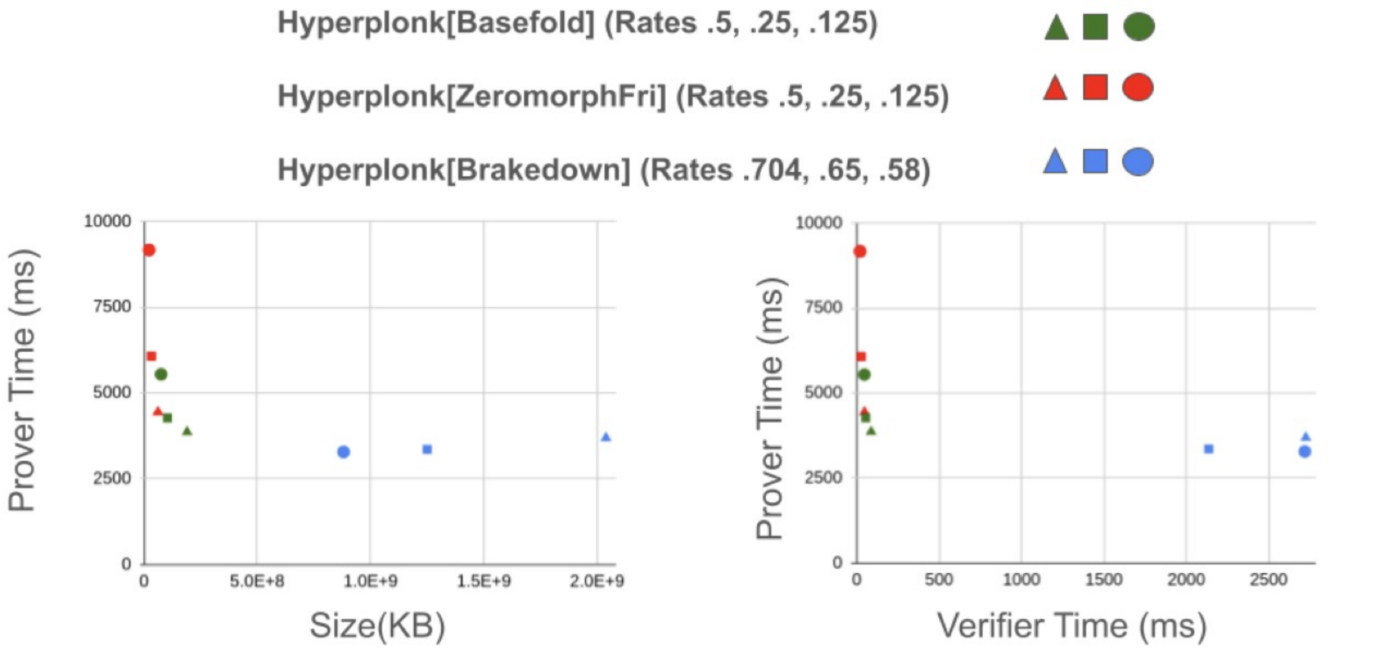
Performance

- Basefold **prover is 2-3 faster** than prior multilinear PCS from FRI when defined over the same finite field
- The Basefold verifier is comparable to FRI's and **~10 times faster than Brakedown's verifier**
- Basefold works over *any sufficiently large finite field*- i.e. proving ECDSA signature verification over secp256k1 is **more than 20 times faster than FRI-based SNARK**

Performance

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

Performance



Performance

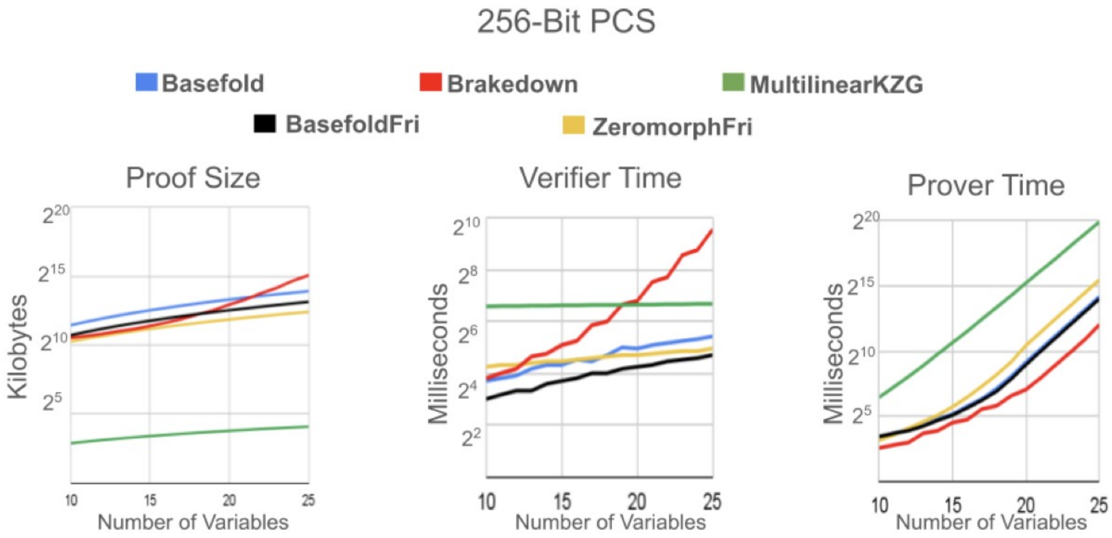


Figure 5: Performance of different PCS over 256-bit fields. Recall that Brakedown and Basefold are field-agnostic while Multilinear-KZG, ZeromorphFri, and BasefoldFri are not.

Future Work

- Explore more practical applications of field-agnosticity
- Prove that Basefold satisfies knowledge soundness even when the verifier checks *within the list-decoding radius* (on-going work)
- Prove better bounds on the distance of the Random Foldable Code/ Find other foldable codes with better distance

Thank you!

Paper: https://eprint.iacr.org/2023/1705.pdf,

Code: https://github.com/hadasz/plonkish_basefold

Contact:

email: hadas.zeilberger@yale.edu

X (twitter): @idocryptography