Blaze: Fast SNARKs from Interleaved RAA Codes

Hadas Zeilberger & Martijn Brehm

Yale University University of Amsterdam

Joint work with Binyi Chen, Ben Fisch, Nicolas Resch, and Ron Rothblum

Blaze

Blaze is a new multilinear polynomial commitment scheme over binary fields with a linear time prover and polylogarithmic verifier costs

Blaze Asymptotics

Cost of committing to $P: \mathbb{F}^m \to \mathbb{F}$ (described by $N = 2^m$ field elements)

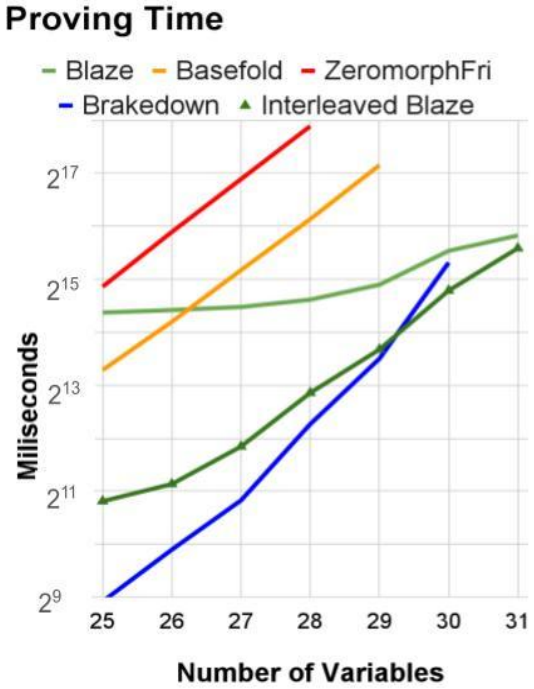
Commitment: $8N$ additions (=XORs) + 1 Merkle Hash

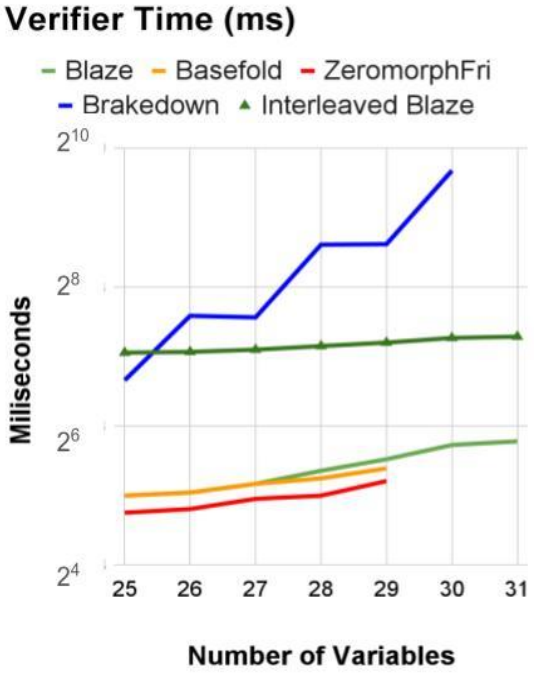
Evaluation proof: dominated by $6N$ additions and $5N$ multiplications

Proof length and Verification: $O_\lambda\left((\log N)^2\right)$

Some Benchmarks

Run on AWS EC2 Instance c6a.48xlarge, with 192 vCPUs and 384GiBs of RAM





Comparison to Prior Works: Concrete

Blaze shines when committing to relatively large multilinears.

For example:

10x faster then Multilinear FRI for $m = 28$ [BBHR18a,BGKS20]

5x faster than Basefold for $m = 29$ [ZCF24]

1.16x slower than Brakedown but 10x smaller proof for $m = 30$

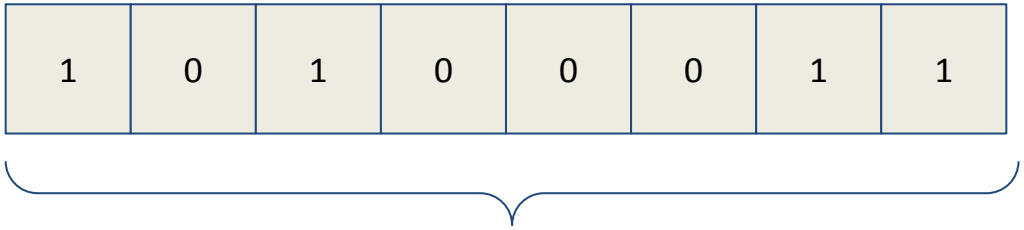
These numbers are for commiting to 64-bit field elements.

Leveraging binary fields lets us commit to bits, which has a **major** saving that is not accounted for here. [DP23,DP24]

Roadmap

- Motivation and Background
- Polynomial Commitment Scheme Construction
- RAA Codes

Why Binary Fields



A single field element is a bit vector

Addition of two field elements is just XOR

Avoid embedding overhead [DP23]

Multiplication is trickier but we don’t do a lot of it

Why Multilinear Polynomials

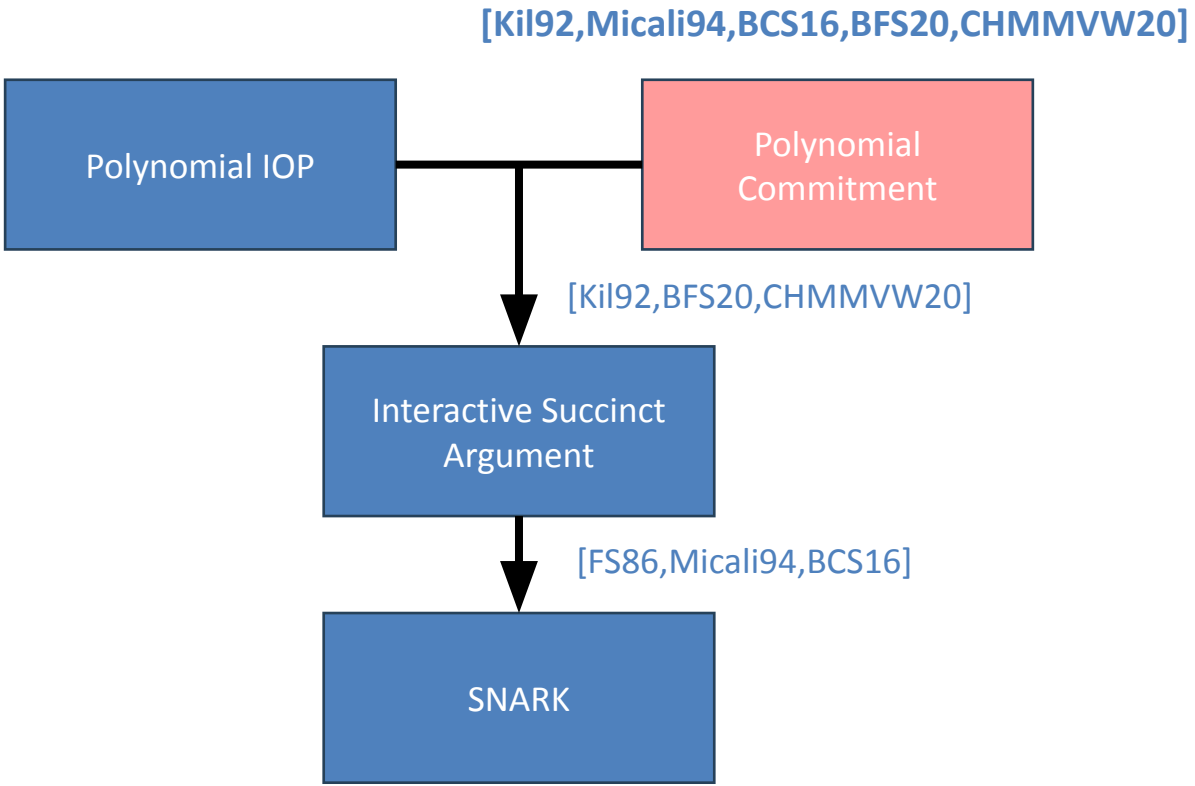
[CBBZ23,GWC19]

- 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	$\mathcal{R}_{\mathrm{R1CS}}$	Ark-Spartan	$\mathcal{R}_{\mathrm{PLONK}+}$	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]

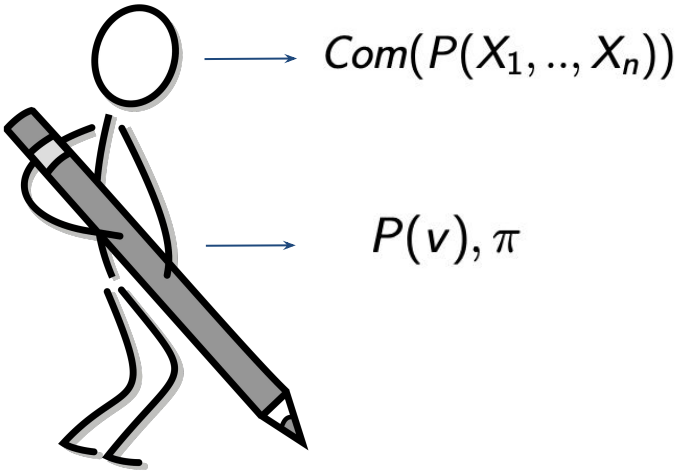
*Jellyfish is an implementation of Plonk

Common SNARK Construction Paradigm



Polynomial Commitment Scheme

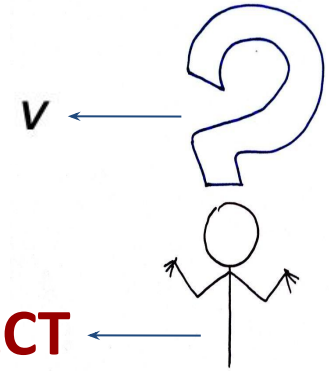
Prover



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

$P(v), \pi$

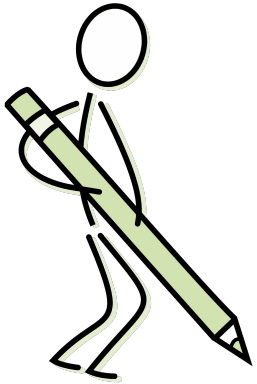
Verifier

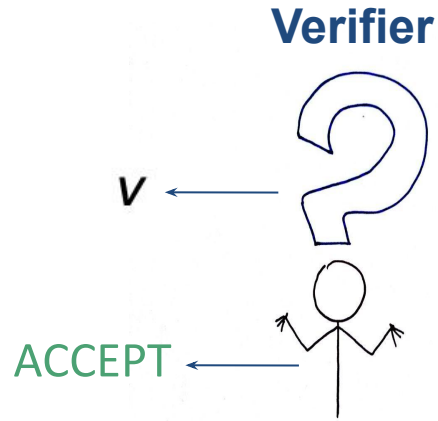


ACCEPT / REJECT

Polynomial Commitment Scheme

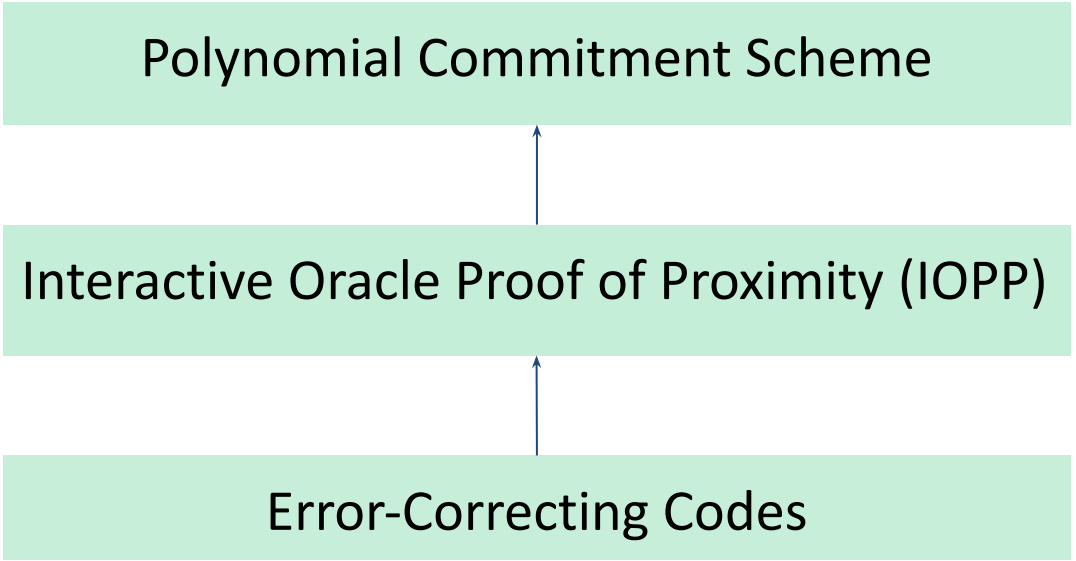
Prover strategy accepted with non-negligible probability….



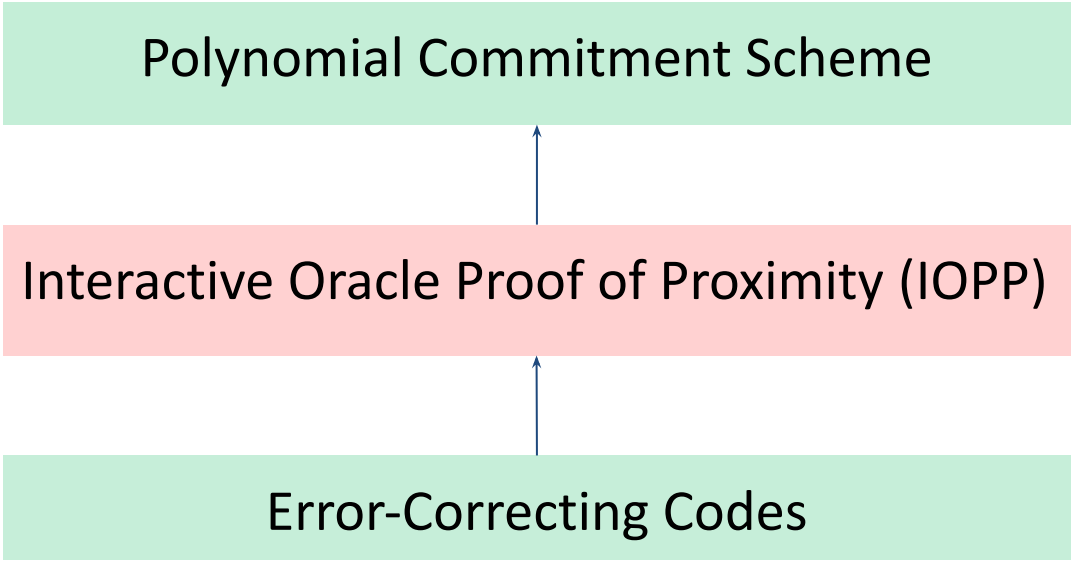


It is possible to efficiently extract a *unique* polynomial from the commitment **(Knowledge Soundness)**

Polynomial Commitment Scheme



Polynomial Commitment Scheme

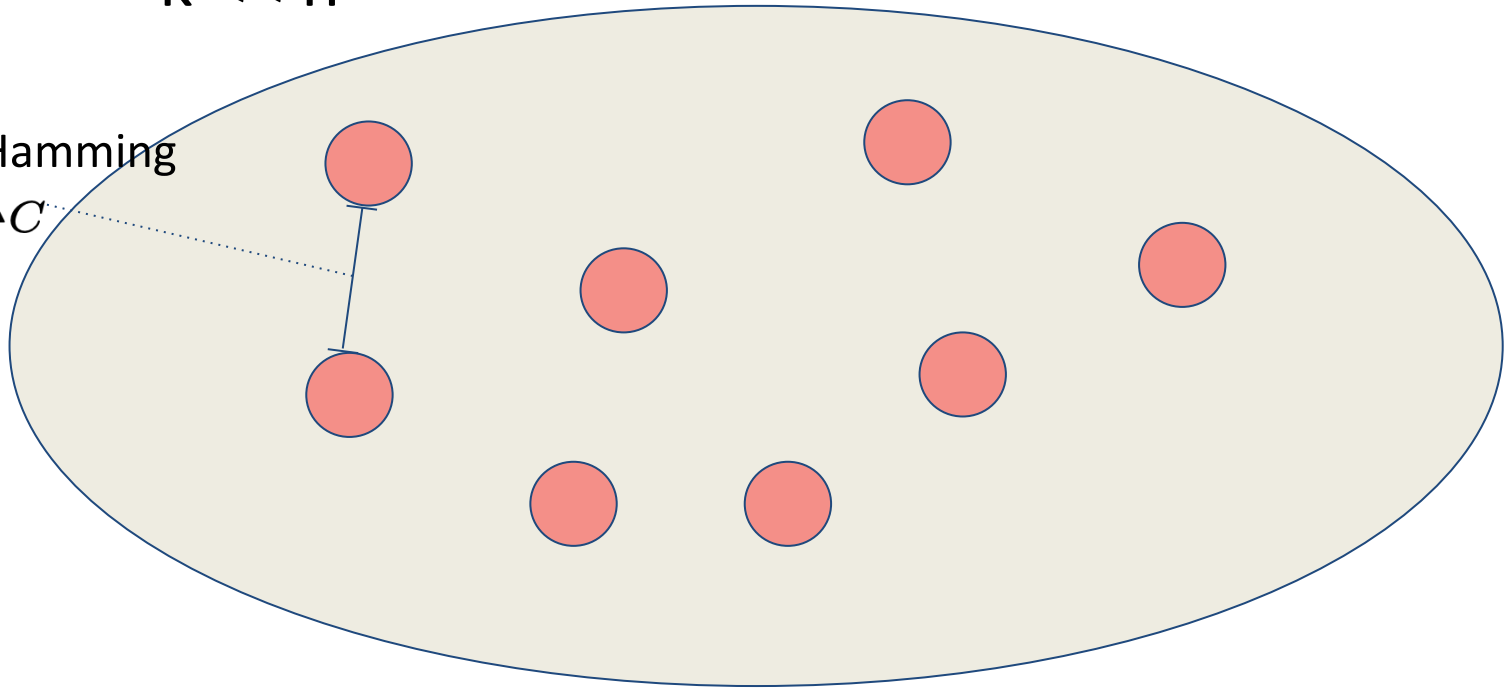


PCS from Error-Correcting Code

A linear error-correcting code is a k-dimensional subspace of $\mathbb{F}^n$

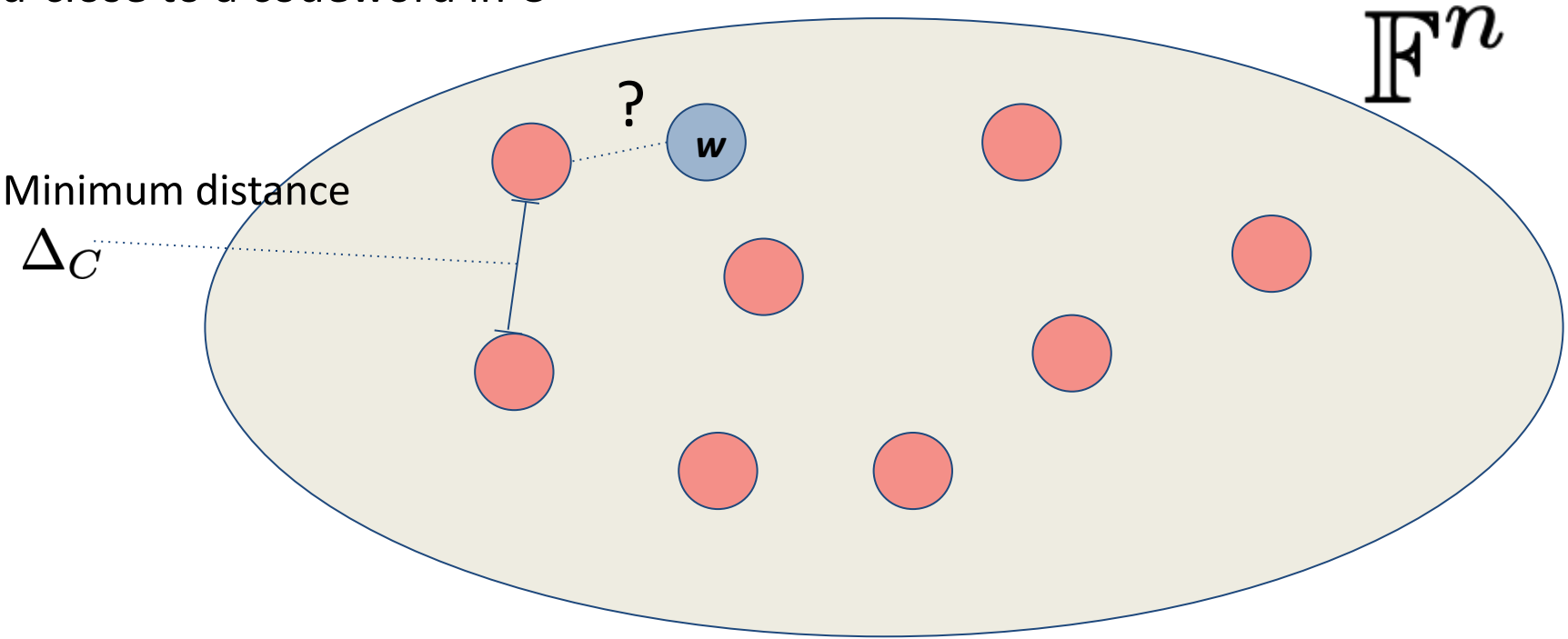
k << n

Minimum Hamming distance Δ_C



PCS from Error-Correcting Code

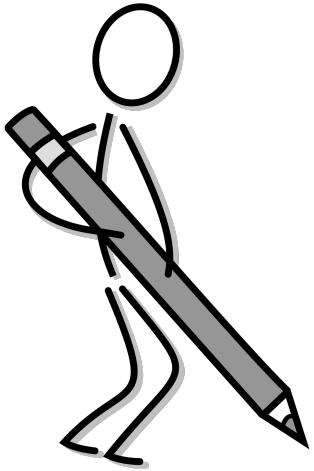
Proximity Proof: Prove relation (d, C; w) : Oracle access to *w*. Prove that is *w* d-close to a codeword in C

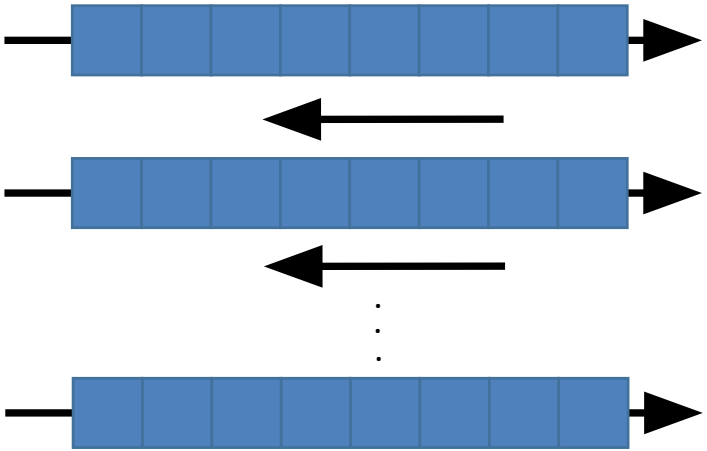


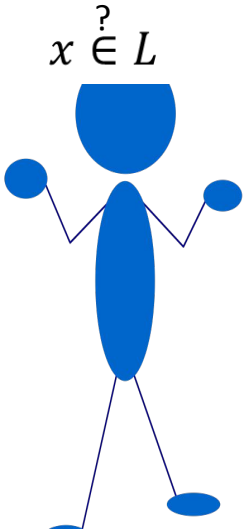
Interactive Oracle Proofs

A hybrid between (public-coin) interactive-proof and PCP.

[BCS16,RRR16]

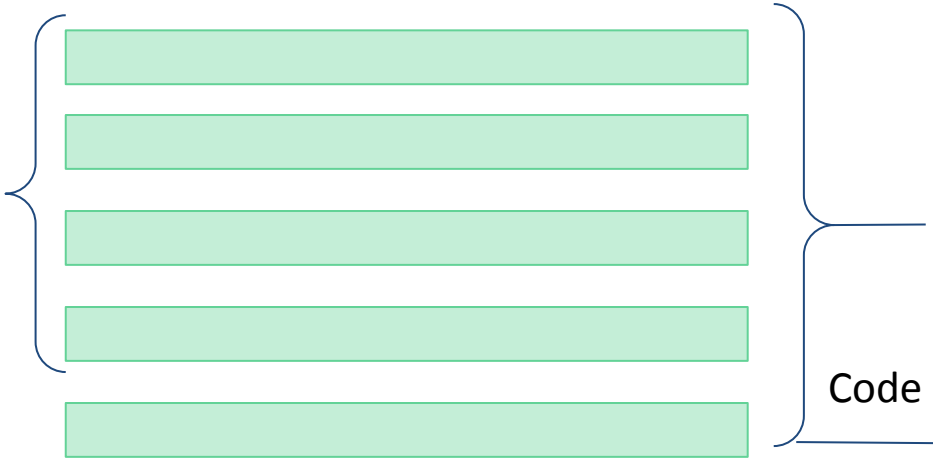






IOPP via Code-Switching

Field Agnostic linear-time IOPP for Interleaved codes [GLSTW23,XZS22,AHIV17]



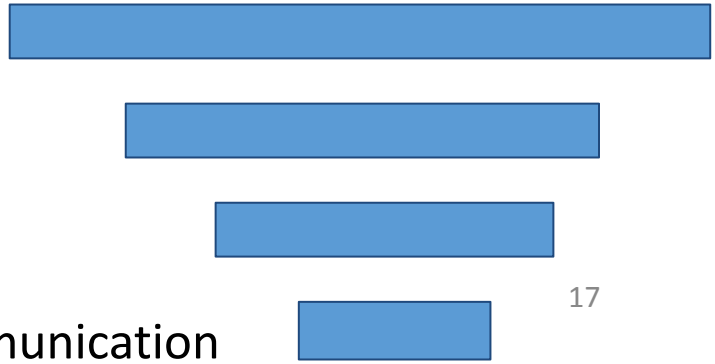
O(sqrt(n)) communication

Better distance -> smaller constant

- [BBHR18a,BGKS20,ZCF24]

Quasilinear IOPP for RS codes

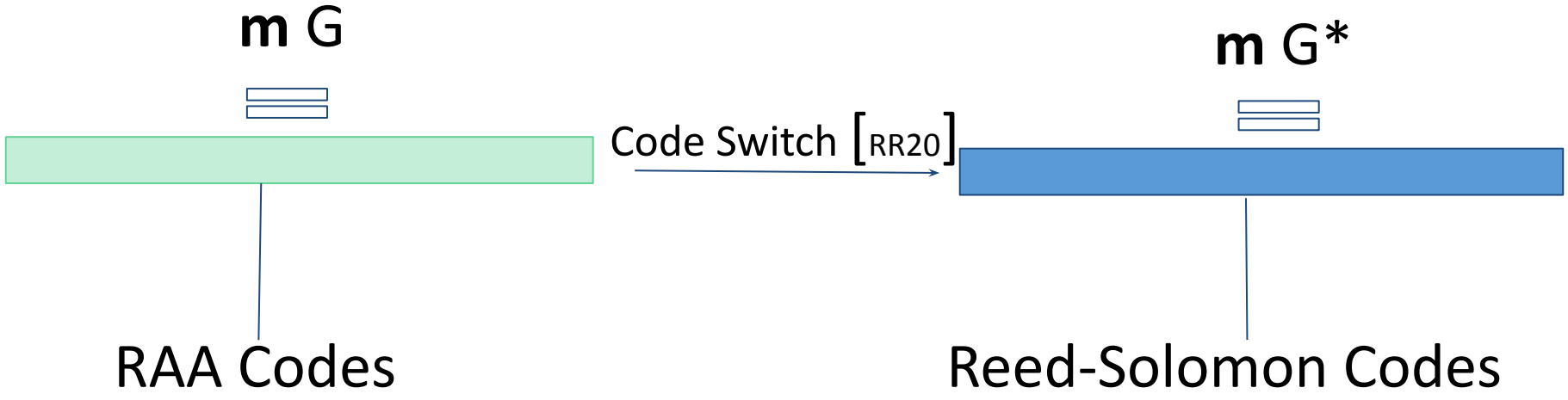
Code Switch [RR20]



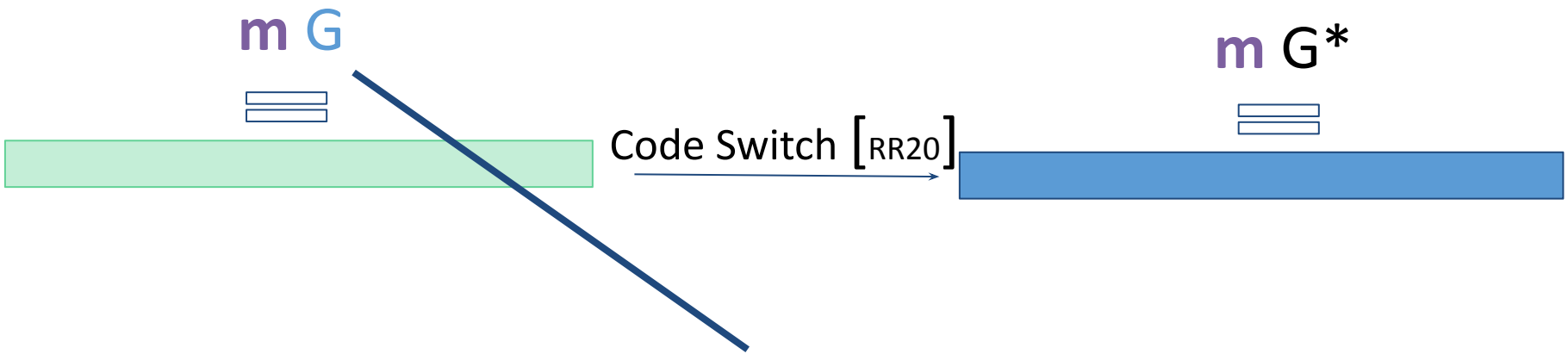
Polylog communication

IOPP via Code-Switching

Prove knowledge of **m** such that

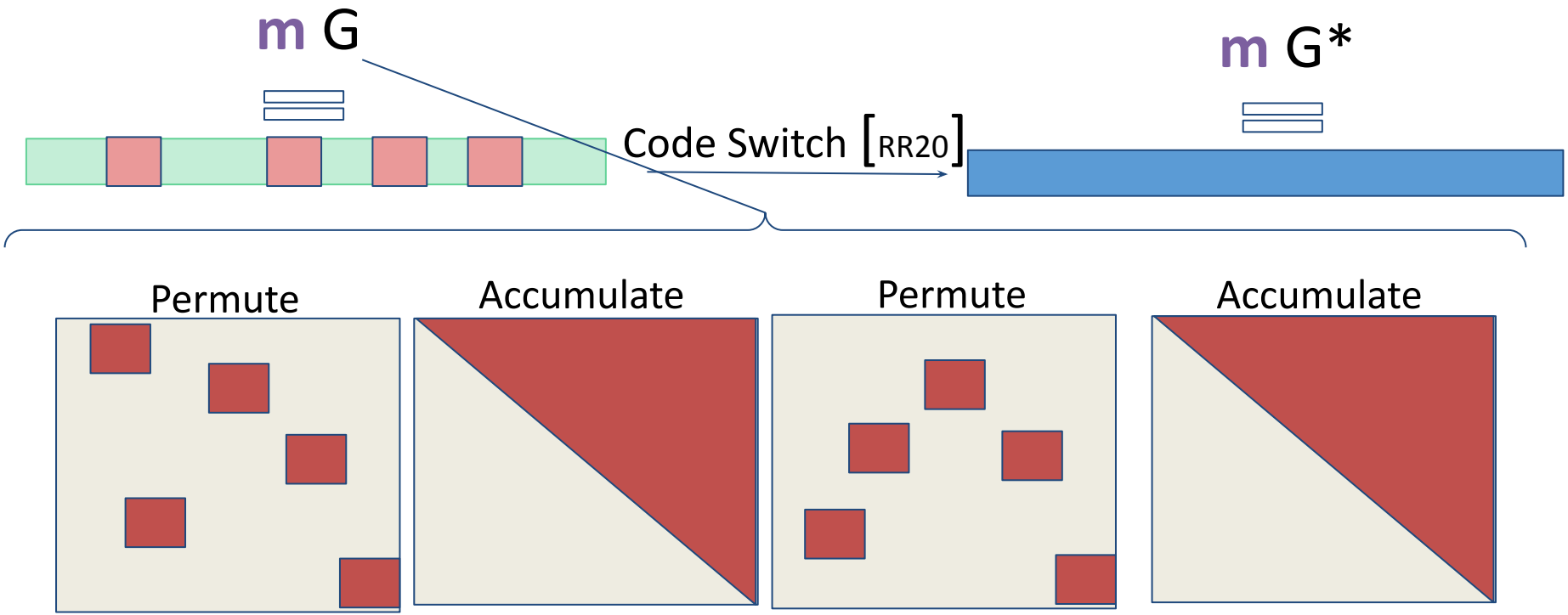


IOPP via Code-Switching

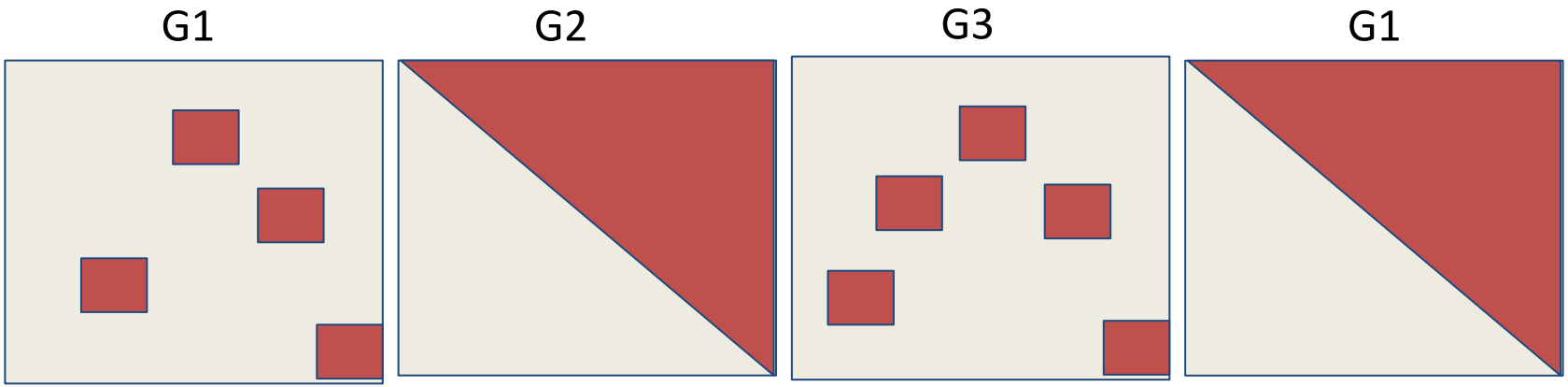


Repeat → Permute → Accumulate → Permute → Accumulate

IOPP via RS -> RAA Code-Switching

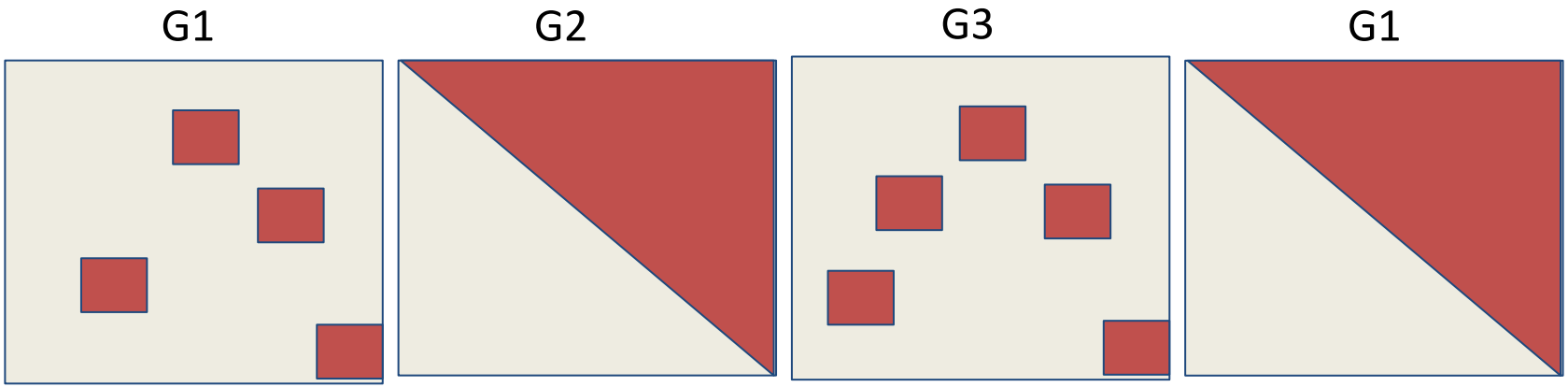


IOPP via RS -> RAA Code-Switching



m G1 = u1 u1 G2 = u2 u2 G3 = u3 u3 G1 = u4

IOPP via RS -> RAA Code-Switching



Permutation Argument [CBBZ24, SL20, THA13]

Accumulation Argument

Permutation Argument

Accumulation Argument

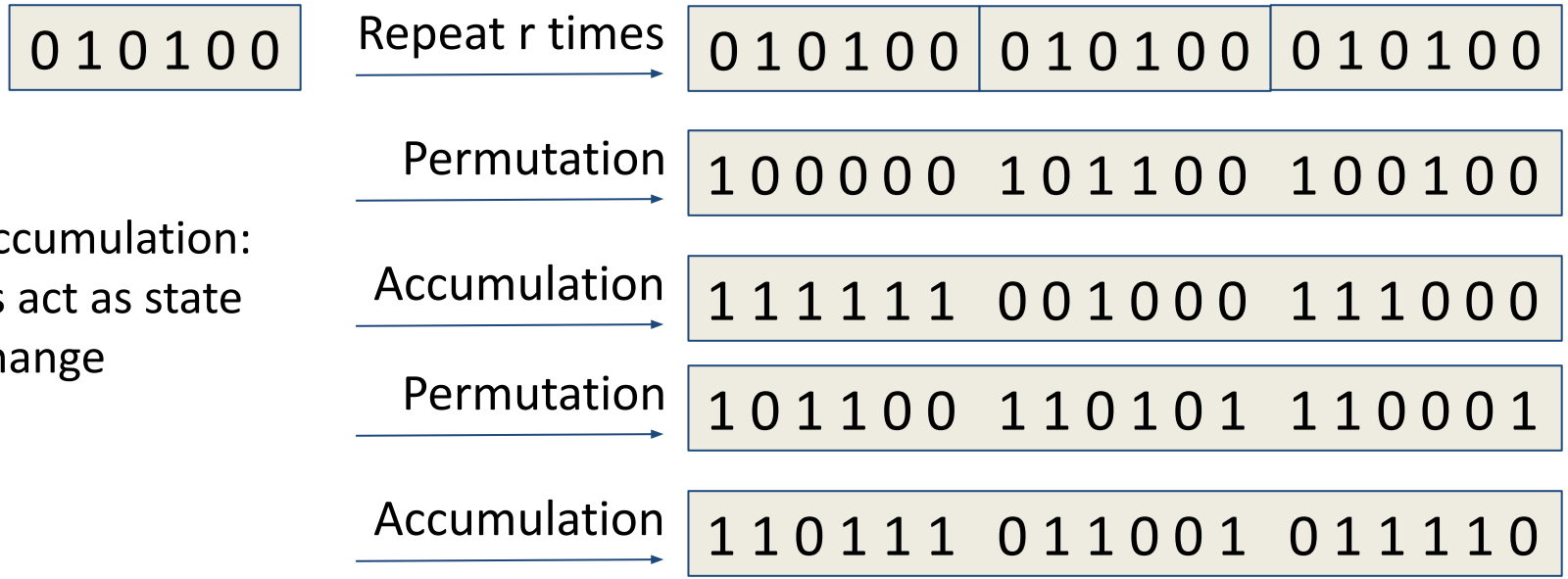
RAA code analysis

Goal: binary error correcting code with good distance (i.e. no low weight vectors) and very efficient encoding map $C : \mathbb{F}_2^{n/r} \rightarrow \mathbb{F}_2^n$

weight(0 1 0 1 0 0) = 2

RAA encoding

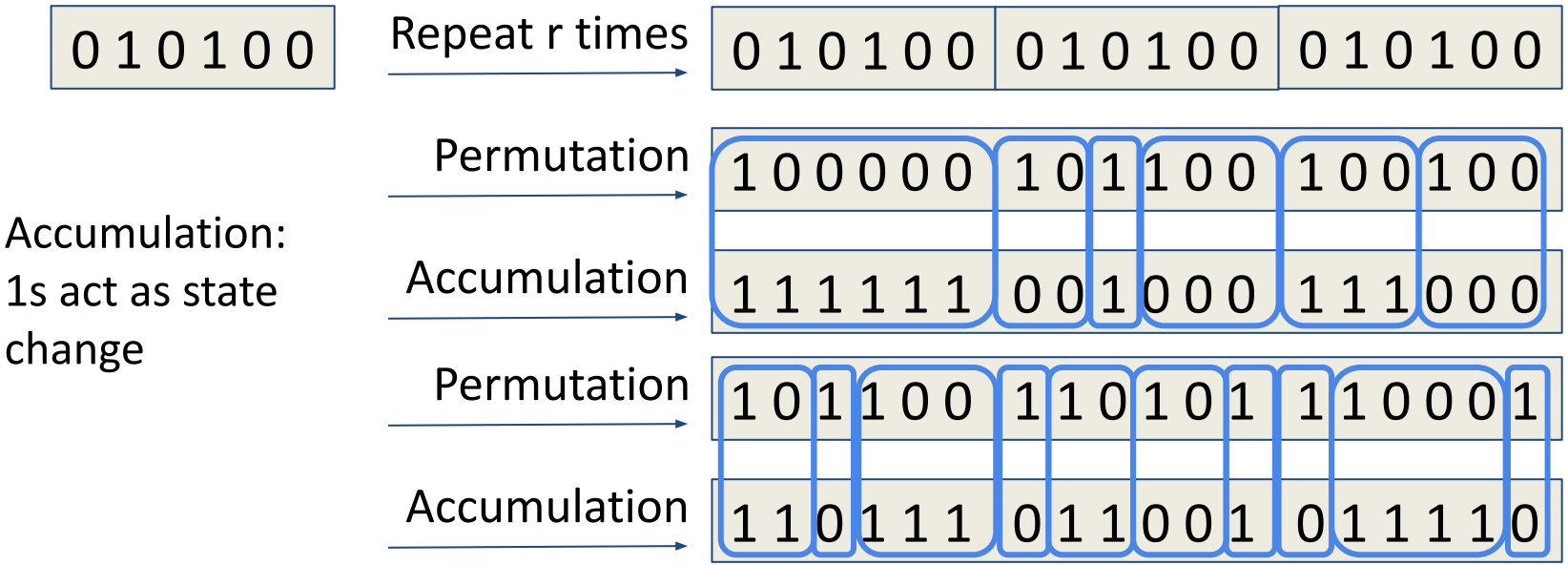
RAA encoding basically as efficient as possible: 2(n-1) XOR gates



Accumulation: 1s act as state change

RAA encoding

RAA encoding basically as efficient as possible: 2(n-1) XOR gates



Accumulation:
1s act as state change

Permute+accumulate increase weight

Why would these operations increase weight?

Accumulate(1 0 0 0 0 1) = 1 1 1 1 1 0

Accumulate(1 1 0 0 0 0) = 1 0 0 0 0 0

Permutation spreads 1s, so it's likely a 1 has some 0s afterwards to flip

Permute+accumulate increase weight

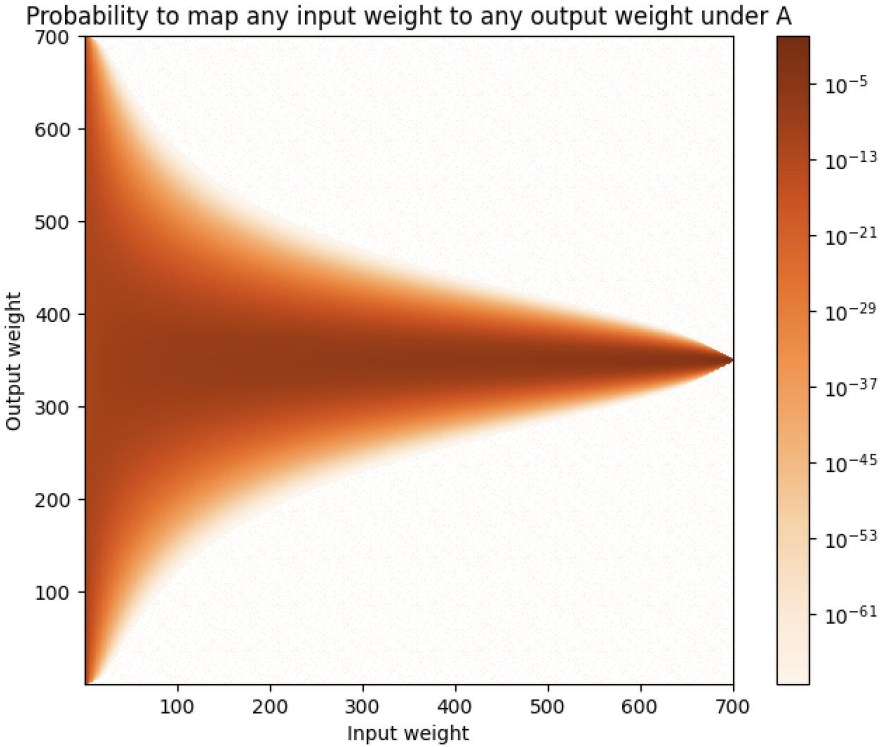
Why would these operations increase weight?

Accumulate(1 0 0 0 0 1) = 1 1 1 1 1 0

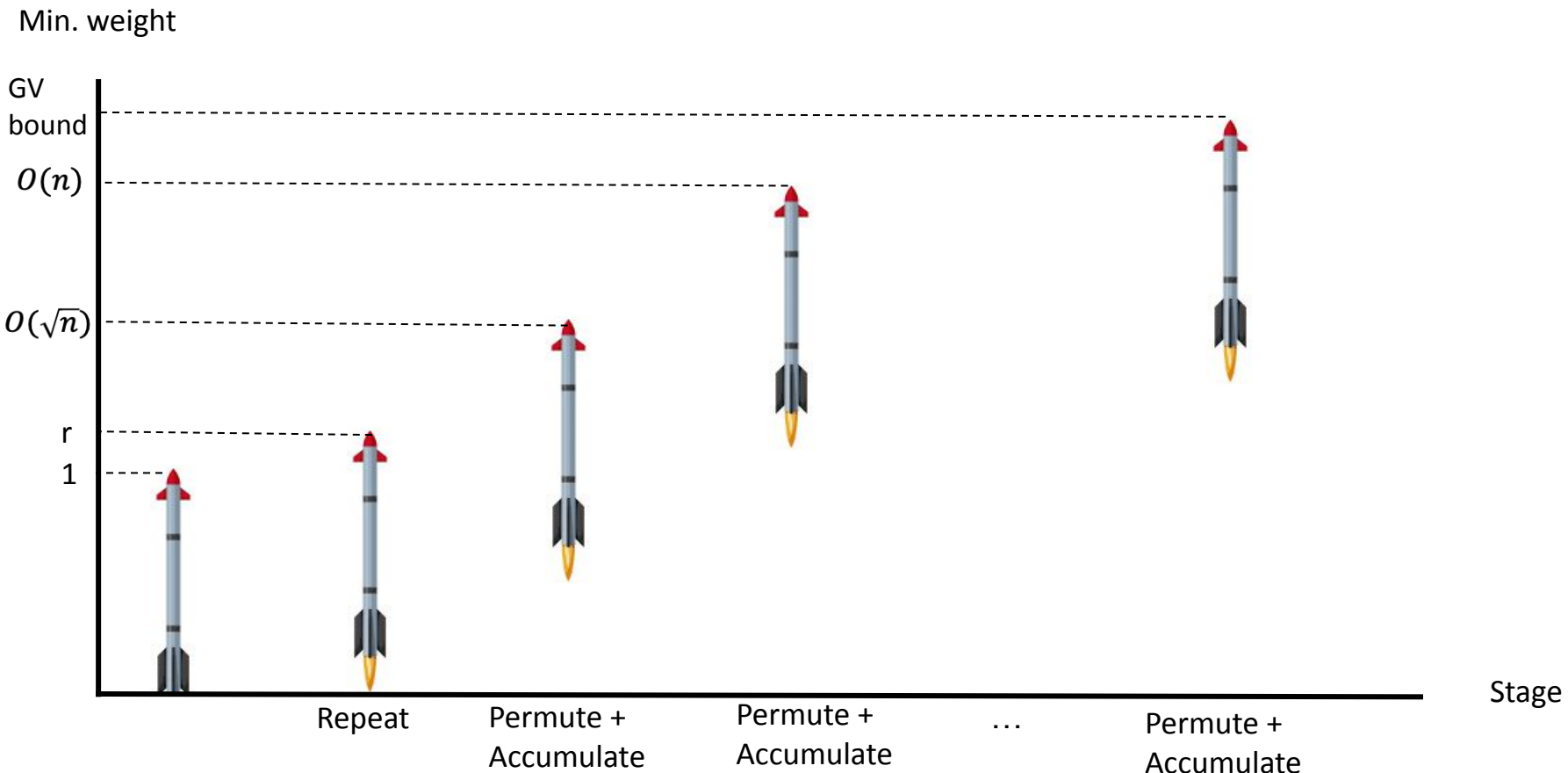
Accumulate(1 1 0 0 0 0) = 1 0 0 0 0 0

Permutation spreads 1s, so it's likely a 1 has some 0s afterwards to flip

After permute+accumulate: expected weight is always n/2



RAA codes: an analogy



RAA distance error probability

Whether RAA code has good distance depends on if you sampled good permutations

Good distance is achieved with probability. Previous work showed that the error probability is o(1) [BMS08,KZCJ07,KZKJ08]

RAA distance error probability

Whether RAA code has good distance depends on if you sampled good permutations

Good distance is achieved with probability. Previous work showed that the error probability is o(1) [BMS08,KZCJ07,KZKJ08]

Question: do we have good distance for, say, n=2^10 or 2^20?

Our techniques build on these works to give inverse polynomial error bounds with concrete constants.

RAA distance error probability

$$\prod_{j=0}^{\frac{r}{2}-1}\frac{(rw_1+2j+1)(rw_1+2j+2)}{(\frac{rw_1}{2}+j+1)^2}\cdot\frac{(w_2-\frac{rw_1}{2}-j)(n-\frac{rw_1}{2}-j)}{(n-w_2-rw_1-2j)(n-rw_1-2j-1)}$$
$$\le\prod_{j=0}^{\frac{r}{2}-1}\frac{(rw_1+2j+2)(rw_1+2j+2)}{(\frac{rw_1}{2}+j+1)^2}\cdot\frac{(w_2-\frac{rw_1}{2}-j)(n-\frac{rw_1}{2}-j)}{(n-w_2-rw_1-2j)(n-rw_1-2j)}$$
$$=\prod_{j=0}^{\frac{r}{2}-1}2^2\cdot\frac{(w_2-\frac{rw_1}{2}-j)(n-w_2-\frac{rw_1}{2}-j)}{(n-rw_1-2j)^2}\,. \tag{12}$$

Now, for fixed j call $x=w_2-\frac{rw_1}{2}-j$ and $y=(n-w_2-\frac{rw_1}{2}-j)$. Observe

$$\frac{(w_2-\frac{rw_1}{2}-j)(n-w_2-\frac{rw_1}{2}-j)}{(n-rw_1-2j)(n-rw_1-2j)}=\frac{xy}{(x+y)^2}=\frac{xy}{(y-x)^2+4xy}\,.$$

Instead of upper bounding this ratio, we find it easier to lower bound its reciprocal:

$$\frac{(y-x)^2+4xy}{xy}=\frac{(y-x)^2}{xy}+4\,.$$

Now, we have $xy\le(w_2-\frac{rw_1}{2})(n-w_2-\frac{rw_1}{2})\le m(n-m)\le mn=n^{1+\gamma}$, as $w_2\le m\le n/2$. On the other hand, $(y-x)=(n-2w_2)\ge(n-2m)\ge\xi n$, where this last inequality uses the assumption $n\ge\frac{2}{1-\xi}m$. Hence,

$$\frac{(y-x)^2}{xy}\ge\frac{\xi^2n^2}{n^{1-\gamma}}=\xi^2n^{1-\gamma}\,.$$

Thus, we have the bound

$$(12)\le\prod_{j=0}^{\frac{r}{2}-1}4\cdot\frac{1}{\xi^2n^{1-\gamma}+4}=\prod_{j=0}^{\frac{r}{2}-1}\frac{1}{(\xi/2)^2n^{1-\gamma}+1}$$
$$\le\prod_{j=0}^{\frac{r}{2}-1}(2/\xi)^2n^{\gamma-1}=(2/\xi)^rn^{\frac{r}{2}(\gamma-1)}=(2/\xi)^rn^{-(1+\varepsilon)}$$

error bounds with

good distance dep

$$(8)\le(10)+\frac{n}{r}\cdot\binom{r}{\frac{r}{2}}\sum_{w_2=1}^{m}\frac{\binom{n-w_2}{d-\lceil w_2/2\rceil}}{\binom{n}{d}}\frac{\binom{n-r}{w_2-r/2}}{\binom{n}{w_2}}\cdot\binom{w_2}{\lceil w_2/2\rceil}\cdot\lceil w_2/2\rceil$$
$$\le(10)+\frac{n}{r}\cdot\binom{r}{\frac{r}{2}}\sum_{v_2=1}^{\lceil\frac{m}{2}\rceil}\frac{\binom{n-2v_2}{d-v_2}}{\binom{n}{d}}\frac{\binom{n-r}{2v_2-r/2}}{\binom{n}{2v_2}}\cdot\binom{2v_2}{v_2}\cdot v_2\,. \tag{15}$$

To bound this expression, we will write out the binomials and then apply the following bound from [KZKJ08]

$$\frac{N^\ell}{\varphi_N(\ell)}\le\prod_{\lambda=0}^{\ell-1}(N-\lambda)\le N^\ell \tag{16}$$

where

$$\varphi_N(\ell):=\exp\left(\frac{\ell(\ell-1)}{2N}\right)\,,$$

which gives us the following:

$$\cdot\binom{r}{\frac{r}{2}}\sum_{v_2=1}^{\lceil\frac{m}{2}\rceil}\frac{(n-2v_2)!\cdot d!\cdot(n-d)!}{n!\cdot(d-v_2)!(n-d-v_2)!}\cdot\frac{(n-r)!(2v_2)!(n-2v_2)!}{n!\cdot(2v_2-\frac{r}{2})!\,(n-2v_2-\frac{r}{2})!}\cdot\binom{2v_2}{v_2}\cdot v_2$$
$$\cdot\binom{r}{\frac{r}{2}}\sum_{v_2=1}^{\lceil\frac{m}{2}\rceil}\frac{\prod_{j=0}^{v_2-1}(d-j)\prod_{j=0}^{v_2-1}(n-d-j)}{\prod_{j=0}^{2v_2-1}(n-j)}$$
$$\cdot\frac{\prod_{j=0}^{\frac{r}{2}-1}(2v_2-j)\prod_{j=0}^{\frac{r}{2}-1}(n-2v_2-j)}{\prod_{j=0}^{r-1}(n-j)}\cdot\binom{2v_2}{v_2}\cdot v_2$$
$$\cdot\binom{r}{\frac{r}{2}}\sum_{v_2=1}^{\lceil\frac{m}{2}\rceil}\frac{d^{v_2}\cdot(n-d)^{v_2}}{n^{2v_2}}\cdot\varphi_n(2v_2)\cdot\frac{(2v_2)^{\frac{r}{2}}(n-2v_2)^{\frac{r}{2}}}{n^r}\cdot\varphi_n(r)\cdot\binom{2v_2}{v_2}\cdot v_2\,.$$

verse polynomial

binomials we keep the $\sqrt{\frac{n}{k(n-k)}}$-factor from Lemma 9.4 to help us out, while for other

we don't bother, as it won't help us much):

$$\binom{r}{v_1}\cdot\frac{\binom{rw_1}{\lceil rw_1/2\rceil}\binom{n-rw_1}{w_2-\lceil rw_1/2\rceil}}{\binom{n}{w_2}}\cdot\frac{\binom{w_2}{\lceil w_2/2\rceil}\binom{n-w_2}{w_3-\lceil w_2/2\rceil}}{\binom{n}{w_3}}\cdot\frac{\lceil rw_1/2\rceil\cdot\lceil w_2/2\rceil}{w_2w_3}$$
$$2^{f(\alpha,\beta,\rho)n}\cdot\frac{102}{100}\cdot\frac{0.61664^3\cdot0.43603^2}{0.33675\cdot0.67352}\cdot\frac{\sqrt{\frac{rw_1}{\lceil rw_1/2\rceil(rw_1-\lceil rw_1/2\rceil)}}\cdot\sqrt{\frac{w_2}{\lceil w_2/2\rceil(w_2-\lceil w_2/2\rceil)}}}{\sqrt{\frac{n}{w_2(n-w_2)}}\cdot\frac{1}{\sqrt{n}}}$$
$$\frac{102}{100}\cdot\frac{0.61664^3\cdot0.43603^2}{0.33675\cdot0.67352}\cdot2^{f(\alpha,\beta,\rho)n}\cdot\sqrt{\frac{4}{rw_1}\cdot\frac{4}{w_2}\cdot\frac{w_2(n-w_2)}{n}}\cdot n$$
$$\frac{102}{100}\cdot\frac{4\cdot0.61664^3\cdot0.43603^2}{0.33675\cdot0.67352}\cdot\frac{1}{\sqrt{r}}\cdot2^{f(\alpha,\beta,\rho)n}\cdot\sqrt{n-w_2}$$
$$\frac{0.80192}{\sqrt{r}}\cdot\sqrt{n}\cdot2^{f(\alpha,\beta,\rho)n}\,.$$

ese two bounds together gives us the proposed bound (18) on the expectation:

$$\sum_{w_1=1}^{n}\sum_{w_2=m+1}^{n}\sum_{w_3=1}^{d}\binom{n/r}{w_1}\cdot\frac{\binom{rw_1}{\lceil rw_1/2\rceil}\binom{n-rw_1}{w_2-\lceil rw_1/2\rceil}}{\binom{n}{w_2}}\cdot\frac{\binom{w_2}{\lceil w_2/2\rceil}\binom{n-w_2}{w_3-\lceil w_2/2\rceil}}{\binom{n}{w_3}}\cdot\frac{\lceil rw_1/2\rceil\cdot\lceil w_2/2\rceil}{w_2w_3}$$
$$\le\sum_{w_1=1}^{n/r}\sum_{w_2=m+1}^{n}\sum_{w_3=1}^{d}\frac{0.80192}{\sqrt{r}}\cdot\sqrt{n}\cdot2^{f\left(\frac{rw_1}{n},\frac{w_2}{n},\frac{w_3}{n}\right)n}$$
$$\le\frac{0.80192\cdot\delta}{\sqrt{r}}\cdot n^{3/2}\sum_{w_1=1}^{n/r}\sum_{w_2=m+1}^{n}2^{f\left(\frac{rw_1}{n},\frac{w_2}{n},\delta\right)n}$$
$$\le\frac{0.80192\cdot\delta}{r^{3/2}}\cdot n^{7/2}\cdot\max_{(\alpha,\beta)\in\mathcal{R}'}2^{f(\alpha,\beta,\delta)n}\,.$$

Note that the second inequality uses the fact that f grows monotonically with ρ for $\rho<1/2$, which we prove below.

RAA distance error probability

We show error probability is roughly $O\left(\frac{1}{n^{1/2}}\right)$

However: most error comes from encoding low weight messages.

Idea: after sampling permutation, check if encoding of low weight messages is high weight (or better: encoding after one “round”).

RAA distance error probability

We show error probability is roughly $O\left(\frac{1}{n^{1/2}}\right)$

However: most error comes from encoding low weight messages.

Idea: after sampling permutation, check if encoding of low weight messages is high weight (or better: encoding after one "round").

Time $O(n^w \log n)$ test decreases error to $O\left(\frac{1}{n^{(1/2)(w+1)}}\right)$

Example: r=4, n=2^20, distance 0.19 (GV bound 0.215):

no test: 2^{-10}	test weight 1: 2^{-20}	test weight 2: 2^{-30}
instant	few secs on laptop	few days on laptop

Some open questions

We can improve error probability to arbitrarily small inverse polynomial, using time scaling as the same polynomial:

$\longrightarrow$ Improve error probability: ideally negligible error in poly-time

There is no analysis of RAA codes over larger alphabets than $\mathbb{F}_2$

We use RAA over $\mathbb{F}_{128}$ in Blaze by simply packing together 128 codewords. This preserves distance, but is inefficient: we could get triple the distance over $\mathbb{F}_{128}$

$\longrightarrow$ Analyze RAA codes over arbitrarily alphabet

Thank you!

Paper: https://eprint.iacr.org/2024/1609.pdf

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

Contact:
Hadas Zeilberger
email: hadas.zeilberger@yale.edu

Contact:
Martijn Brehm
email: m.a.brehm@uva.nl