FRI and Proximity Proofs: what they are, what they are for, and the future

(Part 2)

Dan Boneh
Stanford University

FRI: a Reed-Solomon IOP of Proximity

Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$, $u_0 : \mathcal{L} \to \mathbb{F}$, and $\delta \in [0,1]$

Prover $\boldsymbol{P}(\mathcal{C}, u_0, \cdot)$

Verfier $\boldsymbol{V}^{u_0}(\mathcal{C})$

Goal:

- $u_0 \in \mathrm{RS}[\mathbb{F}, \mathcal{L}, d] \Rightarrow$ Verifier outputs accept (with prob. 1)
- u_0 is δ-far from $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d] \Rightarrow$ Verifier outputs reject w.h.p

We don't care what happens when u_0 is between the two cases

Why needed? A key tool for compiling a Poly-IOP into an IOP.

FRI: a Reed-Solomon IOP of Proximity

Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$, $u_0: \mathcal{L} \to \mathbb{F}$, and $\delta \in [0,1]$

Prover $\boldsymbol{P}(\mathcal{C}, u_0, \cdot)$

Verfier $\boldsymbol{V}^{u_0}(\mathcal{C})$

Recall that:

$\mathcal{L} = \{1, \omega, \omega^2, \ldots, \omega^{n-1}\} \subseteq \mathbb{F}$ and n, d are powers of two,

where $\omega^n = 1$ is an n-th primitive root of unity

Then: $|\mathcal{L}^2| := |\{a^2 : a \in \mathcal{L}\}| = |\mathcal{L}|/2 = n/2$ $\quad$ $(-a,\ a \to a^2)$

$|\mathcal{L}^4| := |\{a^4 : a \in \mathcal{L}\}| = |\mathcal{L}|/4 = n/4$

Review: 2-way folding an arbitrary word $u: \mathcal{L} \to \mathbb{F}$

For $u: \mathcal{L} \to \mathbb{F}$ and $r \in \mathbb{F}$ define $u_e, u_o, u_{\text{fold},r}: \mathcal{L}^2 \to \mathbb{F}$ as

- for $a \in \mathcal{L}$: $\boxed{u_e(a^2) := \frac{u(a)+u(-a)}{2}}$ and $\boxed{u_o(a^2) := \frac{u(a)-u(-a)}{2a}}$
- for $b \in \mathcal{L}^2$: $\boxed{u_{\text{fold},r}(b) := u_e(b) + r \cdot u_o(b)}$

Can similarly define 2^w-way folding for $w \geq 1$.

Recall $|\mathcal{L}^2| = |\mathcal{L}|/2$

Review: an important corollary

Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$ and $\mathcal{C}' = \mathrm{RS}[\mathbb{F}, \mathcal{L}^2, d/2]$

<u>Corollary</u>: For every $u: \mathcal{L} \to \mathbb{F}$ (folding does not decrease distance, w.h.p)

- if $\Delta(u, \mathcal{C}) < 1 - \sqrt{\rho}$ then $\Pr_r\left[\, \Delta\left(u_{\mathrm{fold},r}, \mathcal{C}'\right) \geq \Delta(u, \mathcal{C}) \,\right] \geq 1 - err$
- if $\Delta(u, \mathcal{C}) \geq 1 - \sqrt{\rho}$ then $\Pr_r\left[\, \Delta\left(u_{\mathrm{fold},r}, \mathcal{C}'\right) \geq 1 - \sqrt{\rho} \,\right] \geq 1 - err$

How FRI works

A Reed-Solomon IOP of Proximity (RS-IOPP)

FRI phase 1: commit phase

Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$, $u_0 : \mathcal{L} \to \mathbb{F}$, and $\delta \in [0,1]$

Prover $\boldsymbol{P}(\mathcal{C}, u_0, \cdot)$

Verfier $\boldsymbol{V}^{u_0}(\mathcal{C})$

Phase 1: (commit)

sample $r_1 \leftarrow \mathbb{F}$

honest prover:

$u_1 \coloneqq (u_0)_{\mathrm{fold}, r_1}$

$u_1 : \mathcal{L}^2 \to \mathbb{F}$

$|\mathcal{L}^2| = |\mathcal{L}|/2$

sample $r_2 \leftarrow \mathbb{F}$

honest prover:

$u_2 \coloneqq (u_1)_{\mathrm{fold}, r_2}$

$u_2 : \mathcal{L}^4 \to \mathbb{F}$

$|\mathcal{L}^4| = |\mathcal{L}|/4$

FRI phase 1: commit phase

Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$, $u_0 : \mathcal{L} \to \mathbb{F}$, and $\delta \in [0,1]$

<u>Prover $\boldsymbol{P}(\mathcal{C}, u_0, \cdot)$</u>

<u>Verfier $\boldsymbol{V}^{u_0}(\mathcal{C})$</u>

Phase 1: (commit)

sample $r_1 \leftarrow \mathbb{F}$

honest prover:
$u_1 := (u_0)_{\mathrm{fold}, r_1}$

$u_1 : \mathcal{L}^2 \to \mathbb{F}$

sample $r_t \leftarrow \mathbb{F}$

honest prover:
$u_t := (u_{t-1})_{\mathrm{fold}, r_t}$

$u_t : \mathcal{L}^{2^t} \to \mathbb{F}$

$|\mathcal{L}^{2^t}| = |\mathcal{L}|/2^t$

FRI phase 2: query phase

Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$, $u_0 : \mathcal{L} \to \mathbb{F}$, and $\delta \in [0,1]$

Phase 2: (query)

Verfier $V^{u_0,\dots,u_t}(\mathcal{C}, r_1, \dots, r_t, u_t)$

$u_1 : \mathcal{L}^2 \to \mathbb{F}$

$u_2 : \mathcal{L}^4 \to \mathbb{F}$

$\vdots$

$u_t : \mathcal{L}^{2^t} \to \mathbb{F}$

for $i = 1,2,\dots,t$:

spot check that $u_i = (u_{i-1})_{\mathrm{fold},r_i}$

output yes if $u_t \in \mathrm{RS}[\mathbb{F}, \mathcal{L}^{2^t}, d/2^t]$

[Prover sent Merkle commits to $u_1, \dots, u_t$]

Note: prover sends short u_t to verifier explicitly

(FRI terminates when u_t is "short enough")

FRI phase 2: query phase

Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$, $u_0 : \mathcal{L} \to \mathbb{F}$, and $\delta \in [0,1]$

<u>Phase 2:</u> (query)

<u>Verfier $\boldsymbol{V}^{u_0,\ldots,u_t}(\mathcal{C}, r_1, \ldots, r_t, u_t)$</u>

$u_1 : \mathcal{L}^2 \to \mathbb{F}$

$u_2 : \mathcal{L}^4 \to \mathbb{F}$

$\vdots$

$u_t : \mathcal{L}^{2^t} \to \mathbb{F}$

for $i = 1,2,\ldots,t$:

  spot check that $u_i = (u_{i-1})_{\mathrm{fold},r_i}$

output yes if $u_t \in \mathrm{RS}[\mathbb{F}, \mathcal{L}^{2^t}, {}^{d}/_{2^t}]$

Why is this δ-sound? Intuition: u_0 is δ-far from $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$ $\Rightarrow$

u_1 is "far" from $\mathrm{RS}[\mathbb{F}, \mathcal{L}^2, d/2]$ $\Rightarrow$ … $\Rightarrow$ u_t is "far" from $\mathrm{RS}\left[\mathbb{F}, \mathcal{L}^{2^t}, d/2^t\right]$

How to spot check: method 1

Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$, $u_0 : \mathcal{L} \to \mathbb{F}$, and $\delta \in [0,1]$

Phase 2: (query)

$u_1 : \mathcal{L}^2 \to \mathbb{F}$

$u_2 : \mathcal{L}^4 \to \mathbb{F}$

$\vdots$

$u_t : \mathcal{L}^{2^t} \to \mathbb{F}$

$$z = \frac{u_{i-1}(s) + u_{i-1}(-s)}{2} + \mathrm{r_i} \cdot \frac{u_{i-1}(s) - u_{i-1}(-s)}{2s}$$

Verfier $\boldsymbol{V}^{u_0,\ldots,u_t}(\mathcal{C}, r_1, \ldots, r_t, u_t)$

How to check that $u_i = (u_{i-1})_{\mathrm{fold}, r_i}$:

Repeat m times:

- choose random $s \in \mathcal{L}^{2^{i-1}}$
- query $u_{i-1}(s)$, $u_{i-1}(-s)$, $u_i(s^2)$
- compute $z := (u_{i-1})_{\mathrm{fold}, r_i}(s^2)$
- reject if $z \neq u_i(s^2)$

How to spot check in a picture

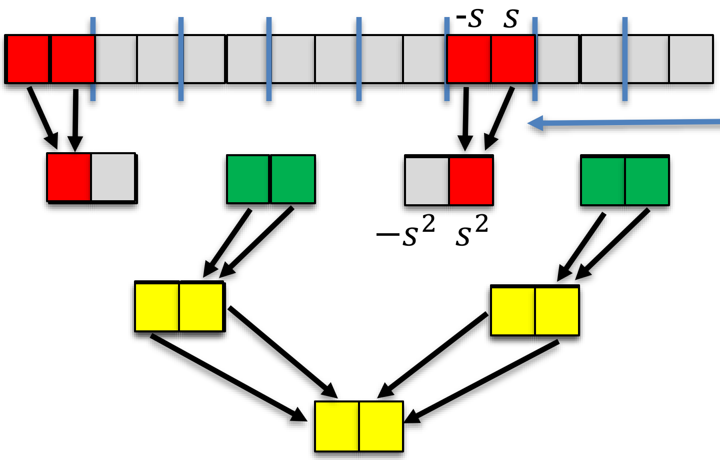
$m = 2$: (spot check at two random spots per oracle)

$u_0 : \mathcal{L} \to \mathbb{F}$

$u_1 : \mathcal{L}^2 \to \mathbb{F}$

$u_2 : \mathcal{L}^4 \to \mathbb{F}$

$u_3 : \mathcal{L}^8 \to \mathbb{F}$



$$u_1(s^2) \stackrel{?}{=} \frac{u_0(s)+u_0(-s)}{2} + r_i \cdot \frac{u_0(s)-u_0(-s)}{2s}$$

Since prover opens $(s, -s)$ jointly, it places both into a single leaf of Merkle tree

$\Rightarrow$ one query to get both

Reject if any spot checks fail

Total of $2mt$ queries to oracles: $2m$ per inner oracle (u_1, u_2).

Why is this protocol sound?

For $i = 0, \ldots, t$:   let

- $\mathcal{C}_i := \mathrm{RS}[\mathbb{F}, \mathcal{L}^{2^i}, d/2^i]$ and

- $\eta_i :=$ (distance of u_i to $\mathcal{C}_i$) $= \Delta(u_i, \mathcal{C}_i) = \min_{w \in \mathcal{C}_i}(\Delta(u_i, w))$

(simplified bound)

Thm: if $0 < \eta_0 < 1 - \sqrt{\rho}$ then Pr[Verifier accepts u_0] $\leq \left(1 - \frac{1}{2}\eta_0\right)^m$

if $\eta_0 \geq 1 - \sqrt{\rho}$ then Pr[Verifier accepts u_0] $\leq \left(1 - \frac{1}{2}(1 - \sqrt{\rho})\right)^m$

(recall: m is the number of spot checks per round, and $\rho :=$ (rate of $\mathcal{C}_i$) $= d/|\mathcal{L}|$)

Why is this protocol sound?

Proof idea: To simplify, let's assume that $m = 1$, $\eta_0 < 1 - \sqrt{\rho}$, and

(*) for all $i = 1, \dots, t$ and $r_i \in \mathbb{F}$: $\Delta((u_{i-1})_{\text{fold},r_i}, \mathcal{C}_i) \geq \Delta(u_{i-1}, \mathcal{C}_{i-1})$

folding does not decrease distance

(note: this only holds w.h.p over $r_i \in \mathbb{F}$ by folding corollary)

Then: $\boxed{\Pr[\text{accept}]} \underset{\text{independent spot checks per round}}{=} \prod_{i=1}^{t} \Pr[\text{not reject in round } i] = \prod_{i=1}^{t} [\underbrace{1 - \Delta(u_i, (u_{i-1})_{\text{fold},r_i})}_{\text{prob. } u_i \text{ is accepted after one spot check}}]$

$\underset{\forall x:\ 1 - x \leq e^{-x} = \exp(-x)}{\leq} \exp(-\sum_{i=1}^{t} \Delta(u_i, (u_{i-1})_{\text{fold},r_i})) \underset{\text{triangular inequality}}{\leq} \exp(-\sum_{i=1}^{t} [\Delta((u_{i-1})_{\text{fold},r_i}, \mathcal{C}_i) - \Delta(u_i, \mathcal{C}_i)])$

$\underset{(*)}{\leq} \exp(-\sum_{i=1}^{t} [\Delta(u_{i-1}, \mathcal{C}_{i-1}) - \Delta(u_i, \mathcal{C}_i)]) = \exp(\eta_t - \eta_0) \overset{\eta_t = 0 \text{ otherwise, Verifier rejects}}{=} \exp(-\eta_0) \leq \boxed{1 - \eta_0/2}$

How to spot check: method 2 (the FRI method)

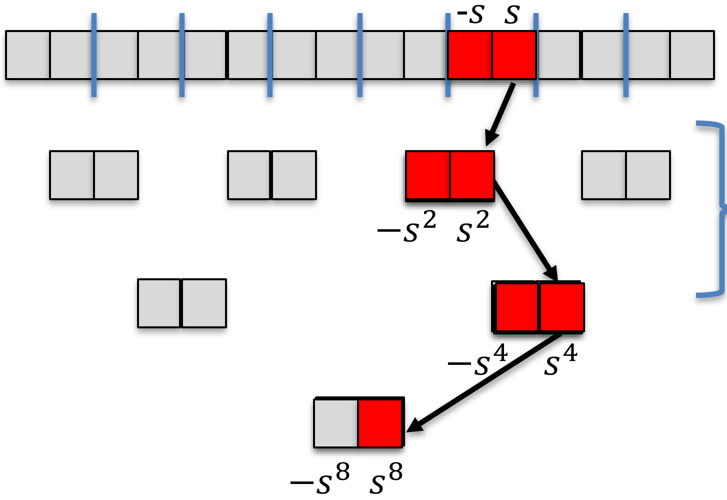
Correlated spot checks: spot check starting at a random $s \in \mathcal{L}$

$u_0 : \mathcal{L} \to \mathbb{F}$

$u_1 : \mathcal{L}^2 \to \mathbb{F}$

$u_2 : \mathcal{L}^4 \to \mathbb{F}$

$u_3 : \mathcal{L}^8 \to \mathbb{F}$



already have $u_1(s^2), u_1(-s^2)$

$\Rightarrow$ only query at $u_2(s^4), u_2(-s^4)$

$\Rightarrow$ 1 query per check

Total of only mt queries to oracles: m per oracle

(recall: method 1 required $2m$ queries per oracle)

How to spot check: the FRI method

Let $\mathcal{C} = \mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$, $u_0 : \mathcal{L} \to \mathbb{F}$

How to check that $u_i = (u_{i-1})_{\text{fold},r_i}$:

Repeat m times:

- choose random $s \in \mathcal{L}$, query $(z_0, y_0) \leftarrow (u_0(s), u_0(-s))$
- for $i = 1, \ldots, t$:
  - set $s \leftarrow s^2 \in \mathcal{L}^{2^i}$
  - compute $z := (u_{i-1})_{\text{fold},r_i}(s)$ from z_{i-1}, y_{i-1}
  - query $(z_i, y_i) \leftarrow (u_i(s), u_i(-s))$
  - reject if $z \neq z_i$

only one query per round

Why is this sound? (using notation as in earlier proof)

[BKS'18, Thm 7.2]

Proof idea: To simplify, let's assume that $m = 1$, $\eta_0 < 1 - \sqrt{\rho}$, and

(∗) for all $i = 1, \dots, t$ and $r_i \in \mathbb{F}$: $\Delta((u_{i-1})_{\mathrm{fold},r_i}, \mathcal{C}_i) \geq \Delta(u_{i-1}, \mathcal{C}_{i-1})$

folding does not decrease distance

Then: $\boxed{\Pr[\text{reject}]} = \Pr\left[\bigcup_{i=1}^{t} (\text{not reject in round } i)\right] = \sum_{i=1}^{t} \Pr[\text{not reject in round } i]$

can be made into a union of <u>disjoint</u> events

$$= \sum_{i=1}^{t} \Delta(u_i, (u_{i-1})_{\mathrm{fold},r_i}) \geq \sum_{i=1}^{t} \left[\Delta((u_{i-1})_{\mathrm{fold},r_i}, \mathcal{C}_i) - \Delta(u_i, \mathcal{C}_i)\right]$$

triangular inequality

$\eta_t = 0$ otherwise, Verifier rejects

$$\underset{(*)}{\geq} \sum_{i=1}^{t} [\Delta(u_{i-1}, \mathcal{C}_{i-1}) - \Delta(u_i, \mathcal{C}_i)] = \eta_0 - \eta_t = \boxed{\eta_0} \quad \Rightarrow \quad \boxed{\Pr[\text{accept}] \leq 1 - \eta_0}$$

Comparing methods 1 vs. 2

The FRI method has better soundness: for $\eta_0 = \Delta(u_0, \mathcal{C}_0) < 1 - \sqrt{\rho}$

- **method 1**: Pr[Verifier accepts u_0] ≤ $\left(1 - \frac{1}{2}\eta_0\right)^m$
- **FRI method**: Pr[Verifier accepts u_0] ≤ $(1 - \eta_0)^m$ (lower prob. ⇒ better bound)

To obtain a SNARK via the BCS'16 compiler we need round-by-round soundness

- **method 1**: independent spot checks ⇒ easy to prove R-by-R soundness
- **FRI method**: correlated spot checks ⇒ harder to prove R-by-R soundness

(see [BJKTTZ'23] for R-by-R analysis of FRI)

The number of spot checks m

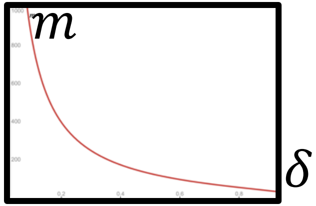
Goal: u_0 is δ-far from $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$ $\Rightarrow$ Pr[Verifier accepts u_0] $\leq 1/2^{128}$

For $\delta < 1 - \sqrt{\rho}$: when u_0 is δ-far from $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$ we know that

Pr[FRI Verifier accepts u_0] $\leq (1 - \delta)^m$

So: we want m where $(1 - \delta)^m \leq 1/2^{128}$

$\Rightarrow$ $m \geq -128/\log_2(1-\delta)$



Main point: (1) the bigger δ is, the smaller m needs to be

(2) smaller m $\Rightarrow$ shorter proof and faster verifier

Choosing the code rate $\boldsymbol{\rho} = d/|\mathcal{L}|$

in practice, set $\delta = \delta_{\max} \approx 1 - \sqrt{\rho}$ (to get smallest possible m)

Example 1: $\rho = 1/2$	Example 2 : $\rho = 1/4$
$\Rightarrow \delta_{\max} \approx 0.29\,, \quad |\mathcal{L}| = 2d$	$\Rightarrow \delta_{\max} \approx 0.5\,, \quad |\mathcal{L}| = 4d$
(Plonky3)	

Proof length: Longer — Shorter (smaller m)

Prover work: Less — More

shorter codewords $\Rightarrow$ less work for Prover to commit

Is 128-bit security enough??

Suppose m is such that Pr[FRI Verifier accepts a **far** u_0] $\leq 2^{-128}$

Fact 1: An adversary that runs FRI 2^{128} times will find a run with favorable spot checks (and forge a proof) with probability $\approx 1/2$

Fact 2: An adversary that runs FRI 2^{80} times will find a run with favorable spot checks (and forge a proof) with probability $\approx 2^{-48}$

For most applications this is sufficient

$\Rightarrow$ do not use less than 120-bits of security; otherwise a 2^{80} adv. will forge proofs.

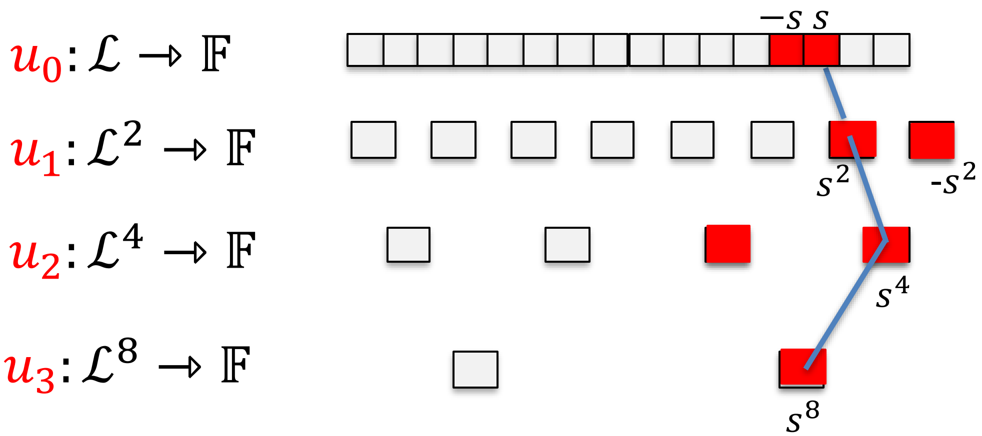
FRI variants

(1) Higher-order folding
(2) Batch FRI for varying degrees
(3) Reduce proof size by grinding
(4) STIR and WHIR variants

(1) The benefits of higher-order folding

$|\mathcal{L}| = n$

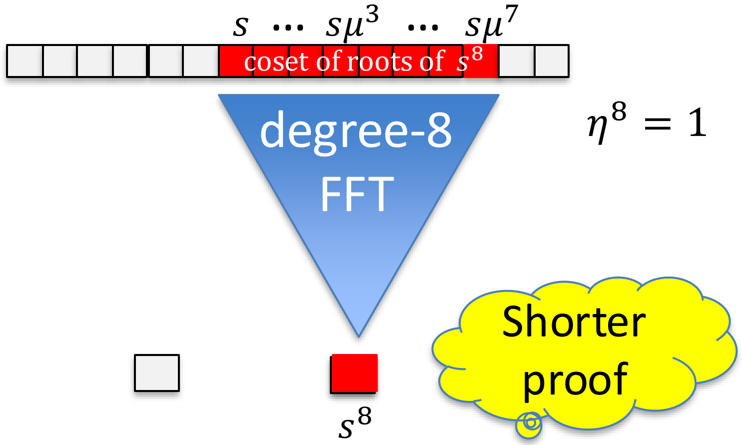
2-way folding **three times**



Protocol has $t = \log_2 n$ rounds

But: total of 6 queries per 8-way step

8-way folding **once**



Protocol has $\log_8 n = t/3$ rounds

vs. total of 8 queries per 8-way step

Can shrink Merkle proofs by placing entire coset in one leaf of Merkle tree:

$\Rightarrow$ 3 Merkle proofs are about $3\log_2 n$ hashes | $\Rightarrow$ 1 Merkle proof is about $\log_2 n$ hashes

(2) Batch FRI for varying degrees: two methods

Poly-IOPP often do multiple RS proximity tests:

Let $u_j: \mathcal{L}_j \to \mathbb{F}$ be words for $j = 1, \ldots, k$

All $u_1, \ldots, u_k$ are encoded with the same rate $\rho = d_j / |\mathcal{L}_j|$

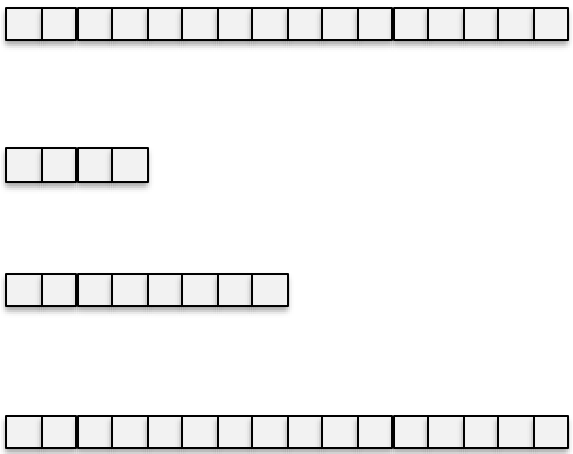
Goal: reject if for any j, u_j is δ-far from $\mathrm{RS}[\mathbb{F}, \mathcal{L}_j, d_j]$

different d_j

The plan: test all k words as a batch.

Method 1: degree padding

Let $d_{max} := \max_j d_j$, $\mathcal{L}_{\max} := \bigcup_j \mathcal{L}_j$



$u_1 \colon \mathcal{L}_1 \to \mathbb{F}$ $d_1 = d_{\max}$ $(= \rho \cdot |\mathcal{L}_{\max}|)$

$u_2 \colon \mathcal{L}_2 \to \mathbb{F}$ $d_2 = d_{\max}/4$

$u_3 \colon \mathcal{L}_3 \to \mathbb{F}$ $d_3 = d_{\max}/2$

$u_4 \colon \mathcal{L}_4 \to \mathbb{F}$ $d_4 = d_{\max}$

Honest prover can interpolate all $u_1, \ldots, u_k$ to $\mathcal{L}_{\max}$

Method 1: degree padding

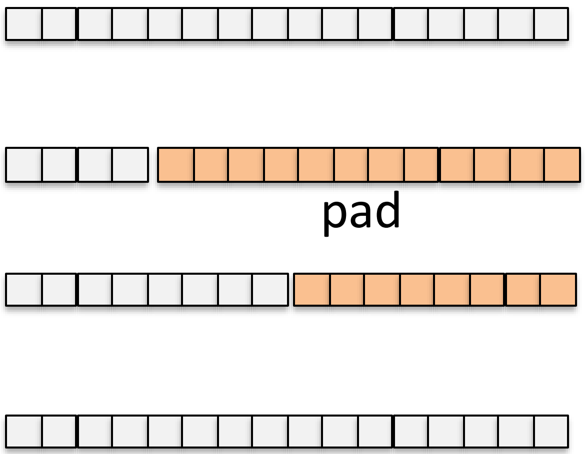
Let $d_{max} \coloneqq \max_j d_j$, $\mathcal{L}_{\max} \coloneqq \bigcup_j \mathcal{L}_j$

$u_1: \mathcal{L}_{\max} \to \mathbb{F}$ $\quad d_1 = d_{\max} \quad (= \rho \cdot |\mathcal{L}_{\max}|)$

$u_2: \mathcal{L}_{\max} \to \mathbb{F}$ $\quad d_2 = d_{\max}/4$

$u_3: \mathcal{L}_{\max} \to \mathbb{F}$ $\quad d_3 = d_{\max}/2$

$u_4: \mathcal{L}_{\max} \to \mathbb{F}$ $\quad d_4 = d_{\max}$



Prover can interpolate all $u_1, \ldots, u_k$ to $\mathcal{L}_{\max}$

Method 1: degree padding

Now, batch as follows: Verifier samples a random $r \in \mathbb{F}$, sends to prover

Honest prover defines: $v_r : \mathcal{L}_{\max} \to \mathbb{F}$ as

$$\boxed{v_r(a) \;\coloneqq\; \sum_{j=0}^{k} \sum_{i=0}^{d_{max}-d_j} r^{e_{i,j}} \cdot [a^i \cdot u_j(a)]}$$

where $e_{i,j}$ is a running counter

Example: suppose $(u_1, d_1), (u_2, d_2), (u_3, d_3)$ s.t. $d_1 = 3$, $d_2 = 4$, $d_3 = 5$.

$$v_r(a) \coloneqq \underbrace{[u_1(a) + r \cdot au_1(a) + r^2 \cdot a^2 u_1(a)]}_{\text{linear comb. of } u_1(a),\ au_1(a),\ a^2 u_1(a)} + \underbrace{[r^3 \cdot u_2(a) + r^4 \cdot au_2(a)]}_{\text{linear comb. of } u_2(a),\ au_2(a)} + r^5 \cdot u_3(a)$$

$u_1 \in \mathrm{RS}[\mathbb{F}, \mathcal{L}_1, 3] \Rightarrow a^2 u_1(a) \in \mathrm{RS}[\mathbb{F}, \mathcal{L}_1, 5]$ $u_2 \in \mathrm{RS}[\mathbb{F}, \mathcal{L}_2, 4] \Rightarrow au_2(a) \in \mathrm{RS}[\mathbb{F}, \mathcal{L}_2, 4]$

Method 1: degree padding

Lemma: [STIR, Lemma 4.13] the transform $(u_1, \ldots, u_k; r) \to v_r$ is distance preserving

case 1: (the honest case)

if $\forall j$: $u_j \in \mathrm{RS}[\mathbb{F}, \mathcal{L}_j, d_\mathrm{j}]$ then $v_r \in \mathrm{RS}[\mathbb{F}, \mathcal{L}_{\max}, d_{\max}]$ for all r.

case 2: (the dishonest case)

if some u_j is δ-far from $\mathrm{RS}[\mathbb{F}, \mathcal{L}_j, d_j]$ then

v_r is δ-far from $\mathrm{RS}[\mathbb{F}, \mathcal{L}_{\max}, d_{\max}]$, w.h.p over r.

The proof follows directly from the RS proximity gap (the BCIKS'20 theorem)

Prover now uses an RS-IOPP to prove that v_r is δ-close to $\mathrm{RS}[\mathbb{F}, \mathcal{L}_{\max}, d_{\max}]$

Method 2: pipelining (no padding or interpolation)

Prover $\boldsymbol{P}(\mathcal{C}, (u_1, u_2, u_3), \cdot)$

Verfier $\boldsymbol{V}^{u_1, u_2, u_3}(\mathcal{C})$

Phase 1: (commit)

suppose $d_2 = d_1/2$ and $d_3 = d_1/4$

sample $r_1 \leftarrow \mathbb{F}$

honest prover:

$$w_1 := (u_1)_{\text{fold}, r_1} \boxed{+ r_1^2 u_2}$$

(fold u_2 into w_1)

$w_1 \colon \mathcal{L}^2 \to \mathbb{F}$

$|\mathcal{L}^2| = |\mathcal{L}|/2$

sample $r_2 \leftarrow \mathbb{F}$

honest prover:

$$w_2 := (w_1)_{\text{fold}, r_2} \boxed{+ r_2^2 u_3}$$

(fold u_3 into w_2)

$w_2 \colon \mathcal{L}^4 \to \mathbb{F}$

$|\mathcal{L}^4| = |\mathcal{L}|/4$

⋮

⋮

(3) Grinding to reduce # of spot checks

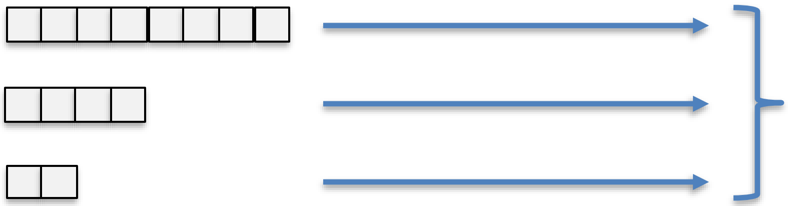
<u>Prover $\boldsymbol{P}((\mathcal{C},\delta),\textcolor{red}{u_0},\cdot)$</u>   <u>Verfier $\boldsymbol{V}^{u_0}((\mathcal{C},\delta))$</u>

Commit phase:

$\textcolor{red}{u_1}: \mathcal{L}^2 \to \mathbb{F}$

$\textcolor{red}{u_2}: \mathcal{L}^4 \to \mathbb{F}$

$\textcolor{red}{u_3}: \mathcal{L}^8 \to \mathbb{F}$



MerkleCommits (including u_0)

Query phase: Derive queries for spot checks by hashing all MerkleCommits (Fiat-Shamir)

<u>Goal</u>: reduce the number of spot checks m (to reduce proof size)

The problem: reducing m below the computed bound enables adversary to try multiple MerkleCommits, until it finds a favorable set of spot checks

Grinding to reduce # of spot checks

One option: add a grinding phase after commit phase (often used in FRI)

Commit phase:

$u_1: \mathcal{L}^2 \to \mathbb{F}$

$u_2: \mathcal{L}^4 \to \mathbb{F}$

$u_3: \mathcal{L}^8 \to \mathbb{F}$

MerkleCommits (including u_0)

Grind: Find G s.t. MSB(SHA3(G,MerkleCommits,nonce)) $= 0^{64}$

Query phase: Derive queries for spot checks by hashing
(Fiat-Shamir) **all MerkleCommits AND G**

Nonce prevents adversary from pre-computing G (e.g, nonce = head of blockchain)

Why does grinding help?

Adversary: wants δ-far u_0 for which it can generate an RS-proximity proof

FRI without grinding:

m is set so that $\mathrm{E}\left[\text{time to find } \delta\text{-far } u_0 \text{ with favorable queries}\right] \geq 2^{128}$

$\Rightarrow$ time to find a false proximity proof is $\approx 2^{128}$

FRI with grinding: every u_0 attempt takes time $\approx 2^{64}$ to find G

$\Rightarrow$ suffice that $\mathrm{E}\left[\text{time to find } \delta\text{-far } u_0 \text{ with favorable queries}\right] \geq 2^{64}$

$\Rightarrow$ can halve the number of spot checks m $\left({}^{128}/_{-\log_2(1-\delta)} \rightarrow {}^{64}/_{-\log_2(1-\delta)}\right)$

$\Rightarrow$ shrink proof length by about $\times 2$

(4) STIR: an FRI variant [ACFY'24]

Recall: in FRI, distances and # spot checks m are fixed round-to-round

input: u_0

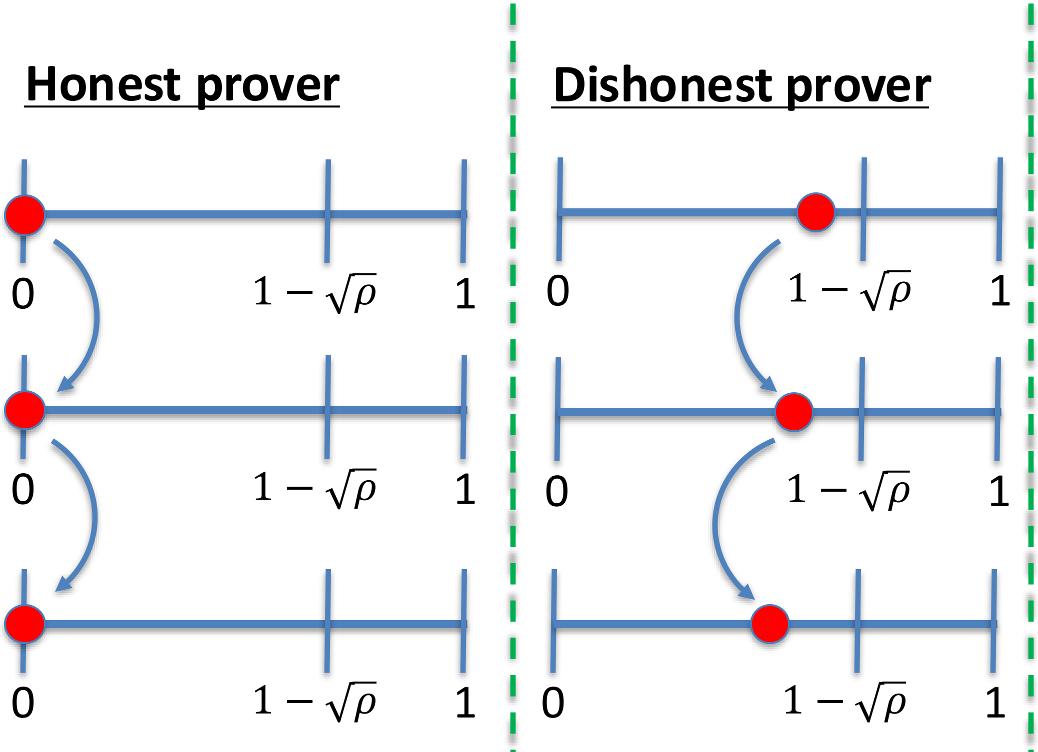
distance u_0 to $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$:

round 1: (4-way fold)

distance u_1 to $\mathrm{RS}[\mathbb{F}, \mathcal{L}^4, d/4]$:

round 2: (4-way fold)

distance u_2 to $\mathrm{RS}[\mathbb{F}, \mathcal{L}^{16}, d/16]$:



spot checks

m

m

STIR: an FRI variant [ACFY'24]

STIR main idea: in each round reduce the rate and increase distance

$\Rightarrow$ # spot checks can be decreased from round to round

Honest prover | **Dishonest prover** | # spot checks

input: u_0

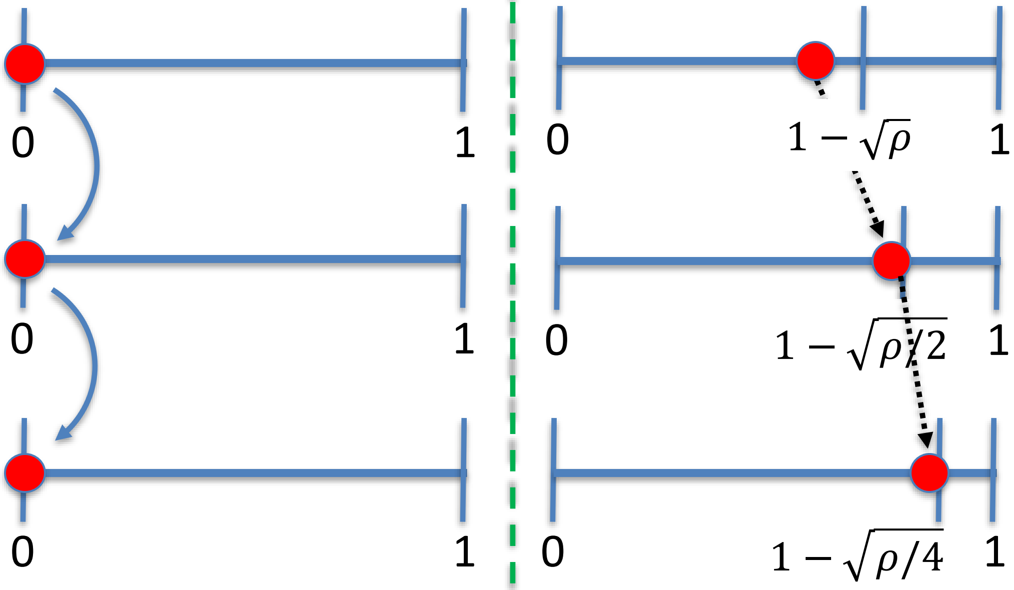
distance u_0 to $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$:
(rate ρ)

round 1: (4-way fold)

distance u_1 to $\mathrm{RS}[\mathbb{F}, \mathcal{L}_1, d/4]$:
(rate $\rho/2$)

round 2: (4-way fold)

distance u_2 to $\mathrm{RS}[\mathbb{F}, \mathcal{L}_2, d/16]$:
(rate $\rho/4$)



m_1

smaller m_2

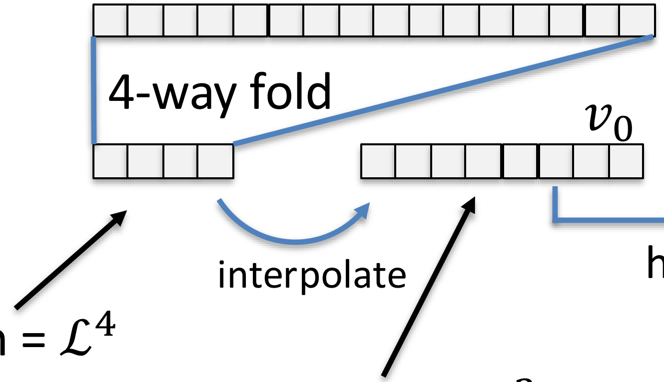
STIR: an FRI variant [ACFY'24]

Idea 1: reduce the code rate by making the honest prover interpolate

Goal: prove u_0 is δ-close to $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$

rate = $\rho = d/|\mathcal{L}|$

$u_0 : \mathcal{L} \to \mathbb{F}$



send $v_0 : \omega\mathcal{L}^2 \to \mathbb{F}$ to Verifier

honest prover interpolates v_0 from $(u_0)_{4\mathrm{fold},r}$

domain = $\mathcal{L}^4$

domain = $\omega\mathcal{L}^2$ (half the size of $\mathcal{L}$)

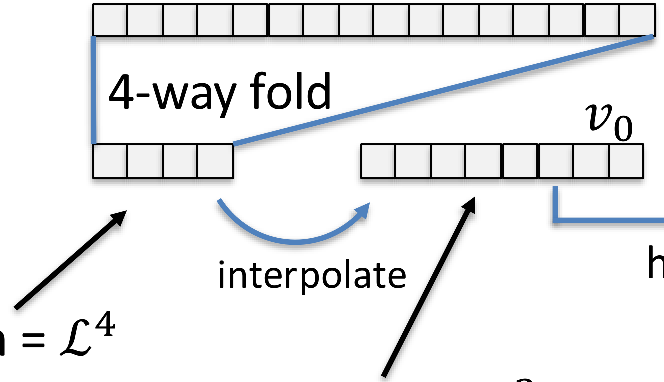
New goal: prove v_0 is δ-close to $\mathrm{RS}[\mathbb{F}, \omega\mathcal{L}^2, d/4]$

rate $= (^d/_4)/(^{|\mathcal{L}|}/_2) = \rho/2 \quad \Rightarrow \quad$ lower rate

STIR: an FRI variant [ACFY'24]

Idea 1: reduce the code rate by making the honest prover interpolate

$u_0 \colon \mathcal{L} \to \mathbb{F}$



domain = $\omega\mathcal{L}^2$ (half the size of $\mathcal{L}$)

Goal: prove u_0 is δ-close to $\mathrm{RS}[\mathbb{F}, \mathcal{L}, d]$

rate = $\rho = d/|\mathcal{L}|$

send $v_0 \colon \omega\mathcal{L}^2 \to \mathbb{F}$ to Verifier

honest prover interpolates v_0 from $(u_0)_{\text{4fold},r}$

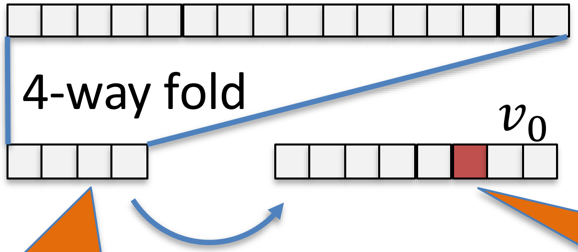
Main point: Lower rate $\rho/2$ means # spot checks for new goal v_0 can be smaller.

Rate drops by a factor of 2 after every folding step $\Rightarrow$ shorter overall proof

STIR: an FRI variant [ACFY'24]

The problem: now we cannot spot check $v_0 : \omega\mathcal{L}^2 \to \mathbb{F}$

$u_0 : \mathcal{L} \to \mathbb{F}$



send $v_0 : \omega\mathcal{L}^2 \to \mathbb{F}$ to Verifier

Can only spot check here

how to spot check this?
Its 4^{th} roots are not in $\mathcal{L}$!!

Idea 2: use quotienting for two things:

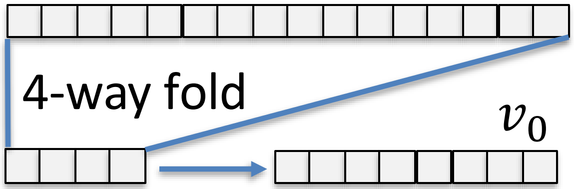
(1) spot checks on v_0 , and

(2) for a malicious prover, increase distance to $\mathrm{RS}[\mathbb{F}, \omega\mathcal{L}^2, d/4]$

How to spot check v_0 by quotienting

How? Honest prover will quotient v_0 by the query points

$u_0 \colon \mathcal{L} \to \mathbb{F}$



<u>Verfier ***V***</u>

send $v_0 \colon \omega\mathcal{L}^2 \to \mathbb{F}$ to Verifier

Verifier sends an out of domain query $t \in \mathbb{F} \setminus \mathcal{L}^4$

Prover sends back $y := f(t)$

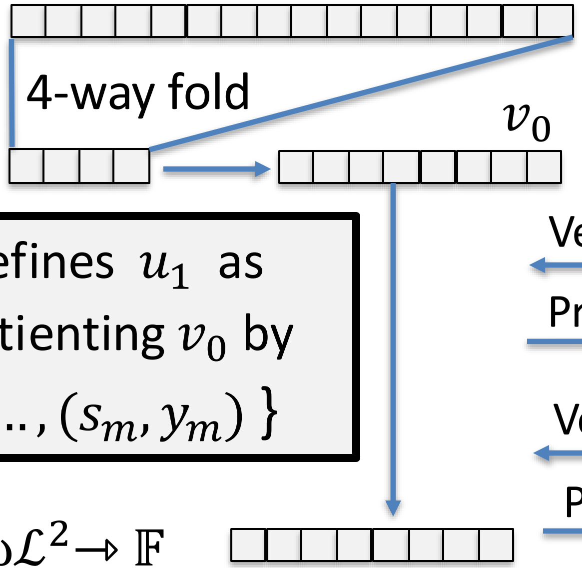
force prover to choose
one $f \in \mathbb{F}^{<d/4}[X]$ s.t.
$\bar{f} \in \mathrm{List}\left[v_0, d/4,\ 1 - \sqrt{\rho/2}\ \right]$

Honest prover will use $f \in \mathbb{F}^{<d/4}[X]$ s.t. $\bar{f} = (u_0)_{4\text{fold},r}$ on $\mathcal{L}^4$

How to spot check v_0 by quotienting

How? Honest prover will quotient v_0 by the query points

$u_0 : \mathcal{L} \to \mathbb{F}$



Honest prover defines u_1 as the result of quotienting v_0 by $\{ (t, y), (s_1, y_1), \dots, (s_m, y_m) \}$

$u_1 : \omega\mathcal{L}^2 \to \mathbb{F}$

<u>Verfier $\boldsymbol{V}$</u>

send $v_0 : \omega\mathcal{L}^2 \to \mathbb{F}$ to Verifier

Verifier sends an out of domain query $t \in \mathbb{F} \setminus \mathcal{L}^4$

Prover sends back $y \coloneqq f(t)$

Verifier sends spot check points $s_1, \dots, s_m \in \mathcal{L}^4$

Prover sends back $y_i \coloneqq f(s_i),\ i = 1, \dots, m$

How to spot check v_0 by quotienting

How? Honest prover will quotient v_0 by the query points

$u_0 : \mathcal{L} \to \mathbb{F}$

4-way fold

v_0

<u>Verfier ***V***</u>

send $v_0 : \omega\mathcal{L}^2 \to \mathbb{F}$ to Verifier

Verifier sends an out of domain query $t \in \mathbb{F} \setminus \mathcal{L}^4$

Prover sends back $y := f(t)$

Verifier sends spot check points $s_1, \ldots, s_m \in \mathcal{L}^4$

Prover sends back $y_i := f(s_i),\ i = 1, \ldots, m$

That is, $u_1(\mathrm{a}) := \frac{v_0(a) - I(a)}{V(a)}$ where

- $I(t) = y$ and $I(s_i) = y_i$
- $V(t) = 0$ and $V(s_i) = 0$

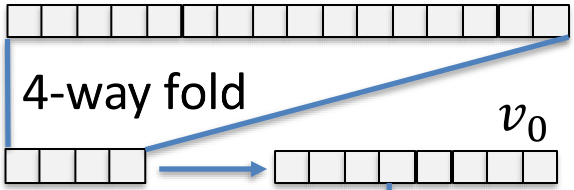
$u_1 : \omega\mathcal{L}^2 \to \mathbb{F}$

In query phase, Verifier computes y_i itself by querying u_0 at $4m$ points and folding

How to spot check v_0 by quotienting

How? Honest prover will quotient v_0 by the query points

$u_0 : \mathcal{L} \to \mathbb{F}$



<u>Verfier ***V***</u>

send $v_0 : \omega\mathcal{L}^2 \to \mathbb{F}$ to Verifier

Verifier sends an out of domain query $t \in \mathbb{F} \setminus \mathcal{L}^4$

Prover sends back $y := f(t)$

Verifier sends spot check points $s_1, \ldots, s_m \in \mathcal{L}^4$

Prover sends back $y_i := f(s_i),\ i = 1, \ldots, m$

That is, $u_1(\mathrm{a}) := \frac{v_0(a) - I(a)}{V(a)}$ where

- $I(t) = y$ and $I(s_i) = y_i$
- $V(t) = 0$ and $V(s_i) = 0$

$u_1 : \omega\mathcal{L}^2 \to \mathbb{F}$

Iterate to prove that u_1 is $(1 - \sqrt{\rho/2}\,)$-close to $\mathrm{RS}[\mathbb{F}, \omega\mathcal{L}^2, d/4]$ with <u>a smaller</u> m

STIR: summary

Main benefit: STIR proof is about $2 \times$ shorter than FRI proof

(using the same rate ρ for the input u_0)

Cons:

- Prover is a bit slower because of interpolation and quotienting
- Verifier is a bit slower because of quotienting
- Batching via pipelining is more cumbersome:
  - Often, functions to batch are all defined using the same rate ρ, but STIR iterations use a different rate in every round

    $\Rightarrow$ Prover will need to interpolate functions to expand to lower rate

WHIR: better than STIR [ACFY'24]

<u>**WHIR**</u>: combines all spot checks into a <u>Sumcheck</u> $\Rightarrow$ no quotienting

- Fold k levels per round, but Verifier now does fewer field ops.

  $\Rightarrow$ fast verifier (≈1.9M gas in the EVM)

- Supports queries to a multilinear polynomial (not just univariate)

How? Not today. (Builds on BaseFold)

The future: other codes

Are there better codes than Reed-Solomon?

The problem with RS-based SNARKs

- **Not field agnostic**: requires an n-th primitive root of unity in $\mathbb{F}$

  $\Rightarrow$ can only use fields where $n = |\mathcal{L}|$ divides $|\mathbb{F}| - 1$

  $\Rightarrow$ difficult to support specific fields (e.g., for ECDSA arithmetic)

- **Encoding is done via an FFT**: takes time $O(n \cdot \log n)$

  $\Rightarrow$ when $n \approx 2^{20}$, the $(\log_2 n)$ causes $20\times$ work for prover

- **FRI enables:** a (univariate Poly-IOP) $\rightarrow$ IOP compiler

  $\Rightarrow$ what about (multilinear Poly-IOP) $\rightarrow$ IOP compiler??

  e.g., $g(x_1, x_2, x_3) = 5x_1 + 2x_2 + 4x_1x_2 + 12x_1x_3 + 7x_1x_2x_3$

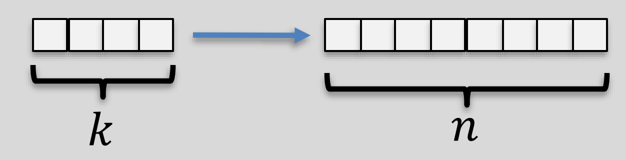
FRI-like proximity proof for other linear codes

FRI can be generalized to any $[n,k,l]_p$ **linear code** $\mathcal{C} \subseteq \mathbb{F}^n$ where:

There is a sequence of linear codes $\mathcal{C} = \mathcal{C}_0,\ \mathcal{C}_1,\ \dots, \mathcal{C}_t$ s.t.

1. $\mathcal{C}_\mathrm{i}$ is a $[n_i,k_i,l_i]_p$ linear code where $n = n_0 > n_1 > \cdots > n_t$ and n_t is “sufficiently small”,
2. there are distance preserving maps $\mathcal{C}_{i-1} \to \mathcal{C}_i$ for $i = 1, \dots, t$,
3. there is a “fast” encoding algorithm $\mathbb{F}^k \to \mathcal{C}$, and
4. min-distance of $\mathcal{C}_0$ is sufficiently large

   (to reduce # of spot checks)



A proximity proof for other linear codes

A **field agnostic** proximity test: (e.g., FRI over the ECDSA prime)

- Gives a (univariate Poly-IOPP) → IOPP over an arbitrary prime p

(1) ECFFT [BCKL'22]:

FRI using functions over an elliptic curve $E/\mathbb{F}_p$,
where the order of $E(\mathbb{F}_p)$ is divisible by n

(even though p is not)

(2) a proximity proof for algebraic geometric codes [BLNR'20].
Here polynomials are replaced with "functions on a curve"

A proximity proof for other linear codes

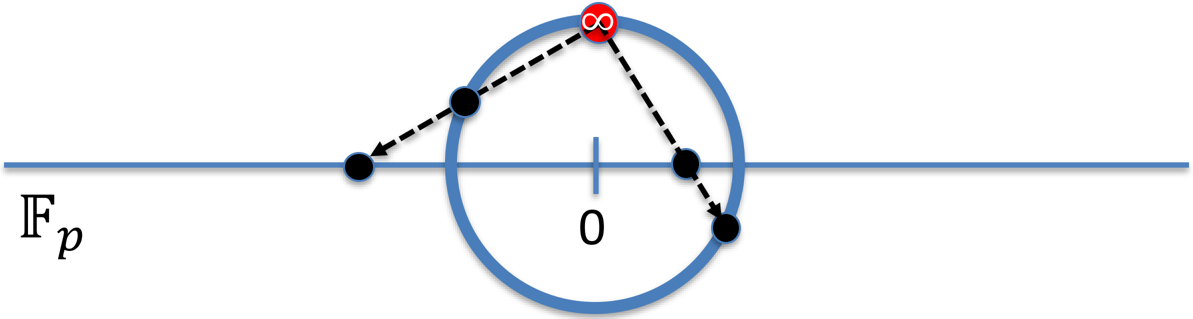
Circle Stark [HLP'24]: Let $M_{11} := 2^{31} - 1$ (the 11th Mersenne prime)

arithmetic mod M_{11} is super fast, ($x \; mod \; M_{11}$ is just an addition)
but $M_{11} - 1$ is not divisible by a high power of 2

Instead: run FRI over the *projective line* mod M_{11} whose size is $M_{11} + 1$

divisible by a high power of 2

One way to represent the projective line is as points on a circle:



So: the circle $(x^2 + y^2 = 1)$ is the same as $\mathbb{F}_p \cup \{\infty\}$ $\Rightarrow$ $(p+1)$ points

BaseFold [ZCF'23]

BaseFold: generalizes FRI to any **foldable** code

⇒ The generalization gives a **field agnostic** proximity test

[note: every foldable code is a multilinear Reed-Muller code]

For a (multilinear Poly-IOPP) → IOPP compiler need a multilinear PCS

- The problem: quotienting only applies to univariates
- BaseFold solution:

  build a multilinear PCS from Sumcheck and a proximity test

(also adopted into Whir)

How? Not today.

More SNARK-useful linear codes

Spielman codes: [BCG'20, Breakdown'21, Orion'22]

- Linear codes with a good minimum distance and a very fast (linear time) encoding algorithm $\mathbb{F}^k \to \mathcal{C}$.
- Also field-agnostic.

Cons: large IOPP proof $\Rightarrow$ large SNARK proof

Expand Accumulate codes: [BFKTWZ'24]

- Field-agnostic codes, but shorter proofs than Breakdown

Cons: $O(n \cdot \log n)$ time encoding.

More SNARK-useful linear codes

Repeat-Accumulate-Accumulate (RAA) codes: [Blaze'24]

- Constructs a multilinear polynomial commitment over $\mathbb{F}_{2^k}$ with a fast (linear time) prover time and $O(\log^2 n)$ proof size

  $\Rightarrow$ $\mathbb{F}_{2^k}$ is friendly to modern CPU instructions

- The commitment uses the tensor code approach of [BCG'20] (making use of Sumcheck).

Much more to do in non-RS based SNARKs

Further reading

- FRI (2018) and analysis (2018): Fast Reed–Solomon Interactive Oracle Proofs of Proximity
- DEEP-FRI (2019): Out of domain sampling improves soundness
- BCIKS (2020): Proximity Gaps for Reed–Solomon Codes
- CircleSTARK (2024): FRI using a Mersenne prime
- STIR (2024): Reed–Solomon proximity testing with fewer queries
- WHIR (2024): Reed–Solomon proximity testing with a fast verifier

Beyond Reed-Solomon codes (a few recent results):

- Breakdown (2021), Orion (2022): Polynomial commitments with a fast prover
- BaseFold (2023): Efficient Polynomial commitments from foldable codes
- Blaze (2024): Fast SNARKs from Interleaved RAA Codes

END OF MODULE