Efficient Proofs of Computational Integrity from Code-Based Interactive Oracle Proofs

Sarah Bordage

Project-team GRACE
LIX, Ecole Polytechnique, Institut Polytechnique de Paris
Inria Saclay Ile-de-France

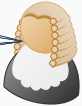
GT GRACE
December 8, 2020

Motivation: Verifiable Computing

Please, run program F on input x for me.

I want to quickly check if your result is correct.

Powerful Prover

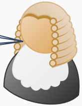


Weak Verifier

Motivation: Verifiable Computing

Please, run program
F on input x for me.

I want to quickly check
if your result is correct.



Powerful Prover

On input (F, x), output result y
and proof of correctness π

y, π

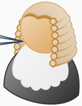
Weak Verifier

On input (F, x, y, π),
accept iff π is a valid proof
for statement "$y = F(x)$"

Motivation: Verifiable Computing

Please, run program F on input x for me.

I want to quickly check if your result is correct.



Powerful Prover

On input (F, x), output result y and proof of correctness π

y, π $\longrightarrow$

Weak Verifier

On input (F, x, y, π), accept iff π is a valid proof for statement "$y = F(x)$"

Our wishlist:

Fast verification

Remark: possible for computations with succinct representation, not for generic circuits, or with pre-processing (setup phase delegated to a trusted party).

No trusted setup

Fast proof generation

Post-quantum security

A view of the "proofs-space" (by crypto assumptions)

	CRHF	DLOG	KoE/AGM/GGM (pairing-based)	Group of unknown order
2013			Pinocchio [PGHR]	
2014			[BCTV]	
2016	ZKBoo [GMO16] SCI [BBC+]	[BCCGP16]	[Groth16] [GM17]	
2017	Ligero [AHIV]	Bulletproof [BBB+] Hyrax [WTS+]	(ZK) vSQL [ZGK+]	
2018	Stark [BBHR] Aurora [BCR+]		vRAM [ZGK+]	
2019	Fractal [COS] Succinct Aurora [BCG+] RedShift [KPV] Virgo [ZXZS]	Spartan [Setty] Halo [BGH]	Sonic [MBK+] Plonk [GWC] Marlin [CHM+] Libra [XZZ+]	Supersonic [BFS]
2020	Virgo++ [ZWZZ]		Mirage [KKPS]	

Some implementations of succinct non-interactive arguments for general computations

PCP-based succinct non-interactive arguments

PCP class

$L \in \mathbf{PCP}[r, q]$ if $\exists$ *efficient* **randomized** $\mathcal{V}$ such that

Completeness: $\forall x \in L, \exists \pi, \mathcal{V}^{\pi}(x) = 1$

Soundness: $\forall x \notin L, \forall \tilde{\pi}, \Pr[\mathcal{V}^{\tilde{\pi}}(x) = 0] > 1/2$

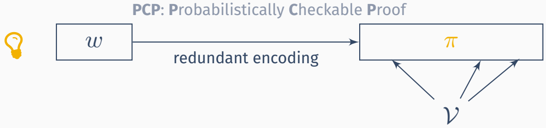
where V reads π at $\leq q$ locations and tosses $\leq r$ coins.

NP class

$L \in \mathbf{NP}$ if $\exists$ *efficient* $\mathcal{V}$ such that:

Completeness: $\forall x \in L, \exists w, \mathcal{V}(x, w) = 1$

Soundness: $\forall x \notin L, \forall \tilde{w}, \mathcal{V}(x, w) = 0$



Starting point: PCP characterization of NP

PCP class

$L \in$ **PCP**$[r, q]$ if $\exists$ *efficient* **randomized** $\mathcal{V}$ such that

Completeness: $\forall x \in L, \exists \pi, \mathcal{V}^{\pi}(x) = 1$

Soundness: $\forall x \notin L, \forall \tilde{\pi}, \Pr[\mathcal{V}^{\tilde{\pi}}(x) = 0] > 1/2$

where V reads π at $\leq q$ locations and tosses $\leq r$ coins.

NP class

$L \in$ **NP** if $\exists$ *efficient* $\mathcal{V}$ such that:

Completeness: $\forall x \in L, \exists w, \mathcal{V}(x, w) = 1$

Soundness: $\forall x \notin L, \forall \tilde{w}, \mathcal{V}(x, w) = 0$

PCP: **P**robabilistically **C**heckable **P**roof

PCP Theorem: **NP** $=$ **PCP**$[\log n, O(1)]$ [BFLS91, FGL+96, ALMSS'98, AS'98,...]

✔ Check **NP** statements way faster than checking an **NP** witness!

✖ PCPs are **not** succinct proofs! ✖ PCP generation is **too expensive**!

30 years later: practical real-world deployment

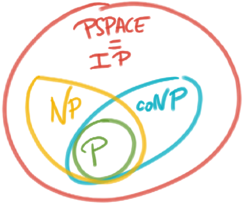
Allow **interaction** with *unbounded* prover $\mathcal{P}$ [Goldwasser-Micali-Rackoff'85, Babai'85]

IP class

$L \in$ **IP** if $\exists \mathcal{V}$ *efficient* **randomized** such that

Completeness: $\forall x \in L, \exists P, \langle P, V\rangle(x) = 1$

Soundness: $\forall x \notin L, \forall \tilde{P}, \Pr[\langle \tilde{P}, V\rangle(x) = 0] > \frac{1}{2}$



Thm: IP = PSPACE [Shamir'86]

Interactive Proofs (IPs) **can be**

- **Zero-knowledge** (ZK): $\mathcal{V}$ learns nothing more than the veracity of the statement.

Assuming the existence of one-way functions, all languages in **NP** have a ZK proof system. [Goldreich-Micali-Wigderson'91]

- **Public-coin:** $\mathcal{V}$ uses only public randomness

Public-coin IPs can be made non-interactive in the Random Oracle Model [Fiat-Shamir'86, Pointcheval-Stern'96]

Succinct interactive arguments for **NP**

[Kilian'92]

Computationally bounded →

Prover

Verifier

Collision-resistant hash function H (CRHF) ←

Compute PCP π proving $x \in L$

Commit(π) →

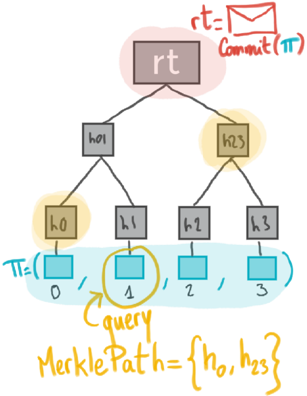
MerkleRoot(π)

random queries q_1, q_2, q_3 ←

Reveal($\pi[q_1]$), Reveal($\pi[q_2]$), Reveal($\pi[q_3]$) →

= ($\pi[q_i]$, MerklePath($\pi[q_i]$))$_i$

Check Merkle paths and run $\mathcal{V}_{\text{PCP}}$



[Kilian'92] First zero-knowledge **sublinear** argument i.e. $O(q \log |\pi|)$

Succinct Non-interactive ARGuments from PCPs

Applying Fiat-Shamir Paradigm to PCPs

[Micali'00]

Computationally bounded $\longrightarrow$

Prover

Same public-randomness

$\longleftrightarrow$

i.e. a CRHF H

Verifier

PCP π for $x \in L$

$h_0 = \text{MerkleRoot}(\pi)$

Derive queries q_1, q_2, q_3 from $H(h_0)$

$p_1 = \text{MerklePath}(\pi[q_1])$

$p_2 = \text{MerklePath}(\pi[q_2])$

$p_3 = \text{MerklePath}(\pi[q_3])$

$$\xrightarrow{\pi = (h_0, \pi[q_1], \pi[q_2], \pi[q_3], p_1, p_2, p_3)}$$

Derive queries,
check Merkle paths
and run $\mathcal{V}_{\text{PCP}}$

- ✔ Succinct argument
- ✔ One message
- ✔ Presumably post-quantum
- ✔ Lightweight crypto
- ✖ PCP generation is **too expensive**

- ✔ Non-interactive in the Random Oracle model ($\rightarrow$ SNARG)
- ✔ Compatible with: Zero-Knowledge, Proof of Knowledge ($\rightarrow$ ZK-SNARK)

Applying Fiat-Shamir Paradigm to PCPs

[Micali'00]

Computationally bounded $\longrightarrow$

Prover

Same public-randomness

$\longleftrightarrow$

i.e. a CRHF H

Verifier

PCP π for $x \in L$

$h_0 = \text{MerkleRoot}(\pi)$

Derive queries q_1, q_2, q_3 from $H(h_0)$

$p_1 = \text{MerklePath}(\pi[q_1])$

$p_2 = \text{MerklePath}(\pi[q_2])$

$p_3 = \text{MerklePath}(\pi[q_3])$

$$\xrightarrow{\pi = (h_0, \pi[q_1], \pi[q_2], \pi[q_3], p_1, p_2, p_3)}$$

Derive queries,
check Merkle paths
and run $\mathcal{V}_{\text{PCP}}$

- ✔ Succinct argument
- ✔ One message
- ✔ Presumably post-quantum
- ✔ Lightweight crypto
- ✖ PCP generation is **too expensive**

- ✔ Non-interactive in the Random Oracle model ($\rightarrow$ SNARG)
- ✔ Compatible with: Zero-Knowledge, Proof of Knowledge ($\rightarrow$ ZK-SNARK)

Applying Fiat-Shamir Paradigm to PCPs

[Micali'00]

Computationally bounded $\longrightarrow$

Prover

Same public-randomness

$\longleftrightarrow$

i.e. a CRHF H

Verifier

PCP π for $x \in L$

$h_0 = \text{MerkleRoot}(\pi)$

Derive queries q_1, q_2, q_3 from $H(h_0)$

$p_1 = \text{MerklePath}(\pi[q_1])$

$p_2 = \text{MerklePath}(\pi[q_2])$

$p_3 = \text{MerklePath}(\pi[q_3])$

$$\xrightarrow{\pi = (h_0, \pi[q_1], \pi[q_2], \pi[q_3], p_1, p_2, p_3)}$$

Derive queries,
check Merkle paths
and run $\mathcal{V}_{\text{PCP}}$

- ✔ Succinct argument
- ✔ One message
- ✔ Presumably post-quantum
- ✔ Lightweight crypto
- ✖ PCP generation is **too expensive**

- ✔ Non-interactive in the Random Oracle model ($\rightarrow$ SNARG)
- ✔ Compatible with: Zero-Knowledge, Proof of Knowledge ($\rightarrow$ ZK-SNARK)

Succinct Non-interactive ARGuments from PCPs

Applying Fiat-Shamir Paradigm to PCPs

[Micali'00]

Computationally bounded $\longrightarrow$

Prover

Same public-randomness

$\longleftrightarrow$

i.e. a CRHF H

Verifier

PCP π for $x \in L$

$h_0 = \text{MerkleRoot}(\pi)$

Derive queries q_1, q_2, q_3 from $H(h_0)$

$p_1 = \text{MerklePath}(\pi[q_1])$

$p_2 = \text{MerklePath}(\pi[q_2])$

$p_3 = \text{MerklePath}(\pi[q_3])$

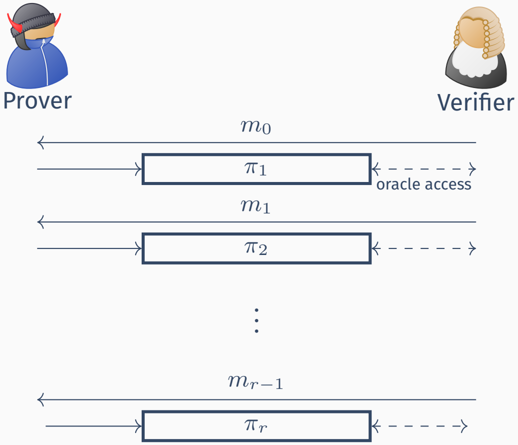
$$\xrightarrow{\pi \,=\, (h_0,\, \pi[q_1],\, \pi[q_2],\, \pi[q_3],\, p_1,\, p_2,\, p_3)}$$

Derive queries,
check Merkle paths
and run $\mathcal{V}_{\mathsf{PCP}}$

- ✔ Succinct argument
- ✔ One message
- ✔ Presumably post-quantum
- ✔ Lightweight crypto
- ✖ PCP generation is **too expensive**

- ✔ Non-interactive in the Random Oracle model ($\rightarrow$ SNARG)
- ✔ Compatible with: Zero-Knowledge, Proof of Knowledge ($\rightarrow$ ZK-SNARK)

IOP Model (**I**nteractive **O**racle **P**roofs)



[BCS16, RRR'16]

IOPs generalize PCPs and IPs

public-coin IOP $\rightarrow$ non-interactive

in the RO model (Fiat-Shamir paradigm)

with communication complexity:

- linear in query complexity of the IOP
- polylog in oracle proof length $|\pi_1| + \ldots + |\pi_r|$

Before: $\underbrace{\text{PCP}}_{\text{information theoretic}}$ + $\underbrace{\text{hash function}}_{\text{crypto}}$ = succinct arguments

From now on: Replace PCPs by IOPs $\rightsquigarrow$ *practical* succinct arguments

From computational integrity to low-degree testing

A computational integrity statement

"z is the result of running program F for T steps."

Verification can be **exponentially faster** than naively re-running the computation.

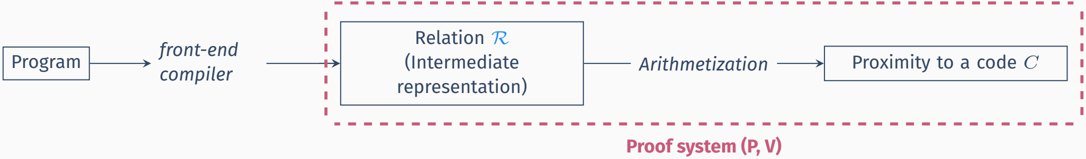
STARK: Scalable **T**ransparent **AR**gument of **K**nowledge [BBHR18]
non-interactive argument for bounded halting problems of a Random-Access Machine (RAM)

Over a finite field $\mathbb{F}$ of cryptographic size:

Setup	Prover	Verifier	Communication complexity	Post-Quantum
Transparent	$O(T \log^2 T)$	$O(\log^2 T)$	$O(\log^2 T)$	yes

Applications:
Allows verification of multiple programs in a single proof (StarkEx, Cairo).
One can build PQ signatures from ZK-STARKs (see Ziggy STARK).

Arithmetization



constraints of a given computation captured by relation $\mathcal{R}$
$\rightarrow$ constraints on low-degree polynomials (e.g. vanish on a given set)
$\rightarrow$ low-degree testing

- If $(x, w) \in \mathcal{R}$, arithmetization produces $c \in C$,
- If $(x, w) \notin \mathcal{R}$, arithmetization produces $\tilde{c}$ which is **very far** from C.

Overview of a STARK: our goal is to construct an IOP $(\mathcal{P}, \mathcal{V})$ with a polylog verifier, logarithmic query complexity, linear oracle proof length, quasilinear prover.

Toy example: a Collatz Sequence

Let's consider a toy example on a **Collatz sequence**.

Start with any positive integer u.
Each term is computed from the previous one as follows:

- if the previous term is even, divide it by 2,
- if the previous term is odd, multiply it by 3 and add 1,

Example: for $u = 42$, it gives (42, 21, 64, 32, 16, 8, 4, 2, 1, 4, 2, 1, ...).

Collatz conjecture: for any positive integer u, the sequence will always reach 1.

Computational integrity statement:

"The Collatz sequence that starts with 42, ends with 1 after 8 iterations."

Collatz sequence: (u_i) defined by $u_0 = u \in \mathbb{N} \setminus \{0\}$ and $u_{i+1} = \begin{cases} u_i/2 & \text{if } u_i \text{ even,} \\ 3u_i + 1 & \text{if } u_i \text{ odd.} \end{cases}$

Computational integrity statement:
"The Collatz sequence that starts with $u = 42$ reaches 1 after $T = 8$ iterations."

Algebraic Intermediate Representation (AIR)
Take $\mathbb{F}$ a large enough prime field.

Witness w_{AIR} (execution trace):

- array $(T+1) \times (a+1)$ of elements in $\mathbb{F}$
- row i: state $\mathbf{S}[i] = (R_0[i], \ldots, R_a[i])$ of the computation at time i
- column j: contents of register R_j over time

Instance x_{AIR}

- **Boundary constraints** e.g. input u, output z
- **Polynomial constraints**
  - $\mathcal{C} \subset \mathbb{F}[\boldsymbol{X}, \boldsymbol{Y}]$, $\mathcal{C} := \{C_0, \ldots, C_p\}$
  - $\boldsymbol{X} = (X_0, \ldots, X_a) \rightsquigarrow$ current state registers
  - $\boldsymbol{Y} = (Y_0, \ldots, Y_a) \rightsquigarrow$ next state registers

AIR relation $\mathcal{R}_{\mathsf{AIR}}$

$$(x_{\mathsf{AIR}}, w_{\mathsf{AIR}}) \in \mathcal{R}_{\mathsf{AIR}} \iff \begin{cases} \text{"input is } u\text{"} \\ \text{"output is } z\text{"} \\ \forall C \in \mathcal{C}, \forall i < T, C(\mathbf{S}[i], \mathbf{S}[i+1]) = 0 \end{cases}$$

An AIR for our toy example

(*) "The Collatz sequence that starts with $u = 42$, ends with 1 after $T = 8$ iterations."

Notation: $\boldsymbol{b} := (2^j)_{0 \le j \le 6}$, $\langle \boldsymbol{b}, S[i] \rangle := \sum_{j=0}^{a} 2^j R_j[i]$ and $\langle \boldsymbol{b}, \boldsymbol{X} \rangle := \sum_{j=0}^{a} 2^j X_i^j$

Witness w_{AIR}: $(T+1) \times (a+1)$ array of elts in $\mathbb{F}$

	R_0	R_1	R_2	R_3	R_4	R_5	R_6	
$i=0$	0	1	0	1	0	1	0	**42**
$i=1$	1	0	1	0	1	0	0	**21**
$i=2$	0	0	0	0	0	0	1	**64**
$i=3$	0	0	0	0	0	1	0	**32**
$i=4$	0	0	0	0	1	0	0	**16**
$i=5$	0	0	0	1	0	0	0	**8**
$i=6$	0	0	1	0	0	0	0	**4**
$i=7$	0	1	0	0	0	0	0	**2**
$i=8$	1	0	0	0	0	0	0	**1**

Instance x_{AIR}:

Boundary constraints

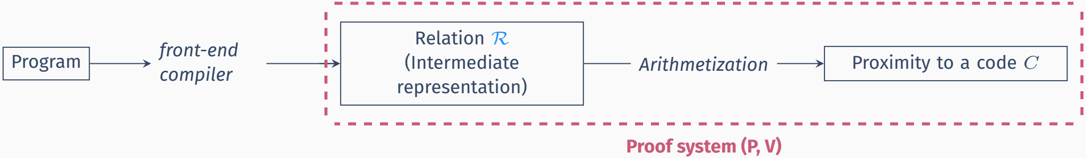
1. $\langle \boldsymbol{b}, S[0] \rangle - 42 = 0$ (first term is 42)
2. $\langle \boldsymbol{b}, S[T] \rangle - 1 = 0$ (last term is 1)

Constraints $\mathcal{C} = \{C_0, \ldots, C_7\} \subset \mathbb{F}[\boldsymbol{X}, \boldsymbol{Y}]$

3. For $j = 0, \ldots, 6$, $C_j(\boldsymbol{X}, \boldsymbol{Y}) = X_j^2 - X_j$
4. $C_7(\boldsymbol{X}, \boldsymbol{Y}) = (1 - X_0)\left(\langle \boldsymbol{b}, \boldsymbol{X} \rangle - 2\langle \boldsymbol{b}, \boldsymbol{Y} \rangle\right)$ $+ X_0 \left(3\langle \boldsymbol{b}, \boldsymbol{X} \rangle + 1 - \langle \boldsymbol{b}, \boldsymbol{Y} \rangle\right)$

$$(x_{\mathsf{AIR}}, w_{\mathsf{AIR}}) \in \mathcal{R}_{\mathsf{AIR}} \iff \begin{cases} \langle \boldsymbol{b}, S[0] \rangle - 42 = 0 \\ \langle \boldsymbol{b}, S[T] \rangle - 1 = 0 \\ \forall C_k \in \mathcal{C}, \forall i < T,\ C_k(\mathbf{S}[i], \mathbf{S}[i+1]) = 0 \end{cases}$$

In the previous episode



✔ constraints of a given computation captured by relation $\mathcal{R}$

$\rightarrow$ constraints on low-degree polynomials (e.g. vanish on a given set)

$\rightarrow$ low-degree testing

- If $(x, w) \in \mathcal{R}$, arithmetization produces $c \in C$,
- If $(x, w) \notin \mathcal{R}$, arithmetization produces $\tilde{c}$ which is **very far** from C.

Step 1: Rational functions which are low-degree polynomials

Assume it exists $g \in \mathbb{F}^\times$ of order $T+1$, $G := \langle g \rangle$.
Let $D \subset \mathbb{F}$ such that $D \cap G = \emptyset$ and $\rho\,|D| = T$. $\rho \in (0,1)$

Reed-Solomon code of dim. k : RS$[\mathbb{F}, D, k] := \{P_{|D} : D \to \mathbb{F} \mid P \in \mathbb{F}[X], \deg P < k\}$.

Encoding the trace (prover's side) For j from 0 to a:

1. Interpolate $P_j(X)$ of degree $\leq T$ such that $P_j(g^i) = R_j[i]$
2. Evaluate $P_j(X)$ on D.

	R_0	R_1	R_2	R_3	R_4	R_5	R_6	
$i=0$	0	1	0	1	0	1	0	**42**
$i=1$	1	0	1	0	1	0	0	**21**
$i=2$	0	0	0	0	0	0	1	**64**
$i=3$	0	0	0	0	0	1	0	**32**
$i=4$	0	0	0	0	1	0	0	**16**
$i=5$	0	0	0	1	0	0	0	**8**
$i=6$	0	0	1	0	0	0	0	**4**
$i=7$	0	1	0	0	0	0	0	**2**
$i=8$	1	0	0	0	0	0	0	**1**

Step 1: Rational functions which are low-degree polynomials

Assume it exists $g \in \mathbb{F}^\times$ of order $T+1$, $G := \langle g \rangle$.
Let $D \subset \mathbb{F}$ such that $D \cap G = \emptyset$ and $\rho\,|D| = T$. $\rho \in (0,1)$

Reed-Solomon code of dim. k : RS$[\mathbb{F}, D, k] := \{P_{|D} : D \to \mathbb{F} \mid P \in \mathbb{F}[X], \deg P < k\}$.

Encoding the trace (prover's side) For j from 0 to a:

1. Interpolate $P_j(X)$ of degree $\leq T$ such that $P_j(g^i) = R_j[i]$
2. Evaluate $P_j(X)$ on D.

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	
g^0	0	1	0	1	0	1	0	42
g^1	1	0	1	0	1	0	0	21
g^2	0	0	0	0	0	0	1	64
g^3	0	0	0	0	0	1	0	32
g^4	0	0	0	0	1	0	0	16
g^5	0	0	0	1	0	0	0	8
g^6	0	0	1	0	0	0	0	4
g^7	0	1	0	0	0	0	0	2
g^8	1	0	0	0	0	0	0	1

Assume it exists $g \in \mathbb{F}^\times$ of order $T+1$, $G := \langle g \rangle$.
Let $D \subset \mathbb{F}$ such that $D \cap G = \emptyset$ and $\rho\,|D| = T$. $\rho \in (0,1)$

Reed-Solomon code of dim. k : RS$[\mathbb{F}, D, k] := \{P_{|D} : D \to \mathbb{F} \mid P \in \mathbb{F}[X], \deg P < k\}$.

Encoding the trace (prover's side) For j from 0 to a:

1. Interpolate $P_j(X)$ of degree $\leq T$ such that $P_j(g^i) = R_j[i]$
2. ~~Evaluate $P_j(X)$ on D.~~

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	
g^0	0	1	0	1	0	1	0	42
g^1	1	0	1	0	1	0	0	21
g^2	0	0	0	0	0	0	1	64
g^3	0	0	0	0	0	1	0	32
g^4	0	0	0	0	1	0	0	16
g^5	0	0	0	1	0	0	0	8
g^6	0	0	1	0	0	0	0	4
g^7	0	1	0	0	0	0	0	2
g^8	1	0	0	0	0	0	0	1

Step 1: Rational functions which are low-degree polynomials

We want to transform $(x_{\text{AIR}}, w_{\text{AIR}})$ into "encoded" counterparts $(x_{\text{RS-AIR}}, w_{\text{RS-AIR}})$.

First, we force the encoded registers to be consistent with the specified input/output.

Instance reduction ($x_{\text{AIR}} \to x_{\text{RS-AIR}}$) [Part 1/2]
Define $(a+1)$ "boundary" polynomials $(B_j(X))_{0\le j\le a}$ such that $\deg B_j < 2$,

$$\underbrace{\left(B_j(g^0)\right)_{0\le j\le a} = (0,1,0,1,0,1,0)}_{\text{input = 42}} \quad \text{and} \quad \underbrace{B_j(g^T) = \begin{cases} 1 \text{ if } j = 0, \\ 0 \text{ otherwise.} \end{cases}}_{\text{output = 1}}$$

and vanishing polynomial $Z_{\text{io}}(X) := (X-1)(X-g^T)$.

Witness reduction ($w_{\text{AIR}} \to w_{\text{RS-AIR}}$) For j from 0 to a:

1. Interpolate $P_j(X)$ of degree $\le T$ such that $P_j(g^i) = R_j[i]$
2. Evaluate $\frac{P_j(X)-B_j(X)}{Z_{\text{io}}(X)}$ on D to get $f_j : D \to \mathbb{F}$ (expected to be a poly of deg $\le T-2$)

$$\begin{cases} P_j(g^0) = B_j(g^0) \\ P_j(g^T) = B_j(g^T) \end{cases} \iff \begin{cases} (X-1) \mid (P_j(X) - B_j(X)) \\ (X-g^T) \mid (P_j(X) - B_j(X)) \end{cases}$$

We want to transform $(x_{\text{AIR}}, w_{\text{AIR}})$ into "encoded" counterparts $(x_{\text{RS-AIR}}, w_{\text{RS-AIR}})$.

First, we force the encoded registers to be consistent with the specified input/output.

Instance reduction $(x_{\text{AIR}} \to x_{\text{RS-AIR}})$ [Part 1/2]
Define $(a+1)$ "boundary" polynomials $(B_j(X))_{0 \le j \le a}$ such that $\deg B_j < 2$,

$$\underbrace{\left(B_j(g^0)\right)_{0 \le j \le a} = (0,1,0,1,0,1,0)}_{\text{input = 42}} \quad \text{and} \quad \underbrace{B_j(g^T) = \begin{cases} 1 \text{ if } j = 0, \\ 0 \text{ otherwise.} \end{cases}}_{\text{output = 1}}$$

and vanishing polynomial $Z_{\text{io}}(X) := (X-1)(X-g^T)$.

Witness reduction $(w_{\text{AIR}} \to w_{\text{RS-AIR}})$ For j from 0 to a:

1. Interpolate $P_j(X)$ of degree $\le T$ such that $P_j(g^i) = R_j[i]$
2. Evaluate $\frac{P_j(X) - B_j(X)}{Z_{\text{io}}(X)}$ on D to get $f_j : D \to \mathbb{F}$ (expected to be a poly of deg $\le T-2$)

If $\mathrm{f} = (f_0, \ldots, f_a)$ **is an encoding of a valid execution trace, then, for any** j, f_j **is a codeword of a code** RS$[\mathbb{F}, D, k]$. (Here, $k = T - 1$)

Step 1: Rational functions which are low-degree polynomials

Witness $w_{\text{RS-AIR}} = (f_0, \ldots, f_j)$ such that $\forall x \in D, f_j(x) = \frac{P_j(x) - B_j(x)}{Z_{\text{io}}(x)}$.

Second, define "rational constraints" on the RS-encoded witness.

Recall AIR's polynomial constraints: $\forall C_k \in \mathcal{C}, \forall i < T,\ C_k(\mathbf{S}[i], \mathbf{S}[i+1]) = 0$.
This means $C_k(P_0(x), \ldots, P_a(x), P_0(gx), \ldots, P_a(gx)) = 0$ for all $x \in \left\{g^i \mid 0 \leq i < T\right\} = G \setminus \left\{g^T\right\}$.

Witness $w_{\text{RS-AIR}} = (f_0, \ldots, f_j)$ such that $\forall x \in D, f_j(x) = \frac{P_j(x) - B_j(x)}{Z_{\text{io}}(x)}$.

Second, define "rational constraints" on the RS-encoded witness.

Recall AIR's polynomial constraints: $\forall C_k \in \mathcal{C}, \forall i < T,\ C_k(\mathbf{S}[i], \mathbf{S}[i+1]) = 0$.
This means $C_k(P_0(x), \ldots, P_a(x), P_0(gx), \ldots, P_a(gx)) = 0$ for all $x \in \left\{g^i \mid 0 \le i < T\right\} = G \setminus \left\{g^T\right\}$.

Idea: Define $Z_G(X) := \prod_{h \in G}(X - h)$. Then,

$$\frac{(X - g^T)}{Z_G(X)} C_k\big((P_0(X), \ldots, P_a(X), P_0(gX), \ldots, P_a(gX)\big) \text{ must be a polynomial.}$$

Step 1: Rational functions which are low-degree polynomials

Witness $w_{\text{RS-AIR}}$ = $(f_0, \ldots, f_j)$ such that $\forall x \in D$, $f_j(x) = \frac{P_j(x) - B_j(x)}{Z_{\text{io}}(x)}$.

Second, define "rational constraints" on the RS-encoded witness.

Recall AIR's polynomial constraints: $\forall C_k \in \mathcal{C}, \forall i < T$, $C_k(\mathbf{S}[i], \mathbf{S}[i+1]) = 0$.
This means $C_k(P_0(x), \ldots, P_a(x), P_0(gx), \ldots, P_a(gx)) = 0$ for all $x \in \left\{g^i \mid 0 \leq i < T\right\} = G \setminus \left\{g^T\right\}$.

Idea: Define $Z_G(X) := \prod_{h \in G}(X - h)$. Then,

$$\frac{(X - g^T)}{Z_G(X)} C_k\big((P_0(X), \ldots, P_a(X), P_0(gX), \ldots, P_a(gX)\big) \text{ must be a polynomial.}$$

We don't have access to P_j directly. But on D, it can be expressed with f_j, B_j and Z_{io}!

Instance reduction ($x_{\text{AIR}} \to x_{\text{RS-AIR}}$) [Part 2/2]

Let $\mathbf{f} = (f_0, \ldots, f_a) \in (\mathbb{F}^D)^{a+1}$. For each $C_k \in \mathcal{C}$, define $C_k[\mathbf{f}] : D \to \mathbb{F}$ s.t. for all $x \in D$:

$$C_k[\mathbf{f}](x) = \frac{(x - g^T)}{Z_G(x)} C_k\big((f_0 Z_{\text{io}} + B_0)(x), \ldots, (f_a Z_{\text{io}} + B_a)(x), (f_0 Z_{\text{io}} + B_0)(gx), \ldots, (f_a Z_{\text{io}} + B_a)(gx)\big)$$

Witness $w_{\text{RS-AIR}}$ = $(f_0, \ldots, f_j)$ such that $\forall x \in D$, $f_j(x) = \frac{P_j(x) - B_j(x)}{Z_{\text{io}}(x)}$.

Second, define "rational constraints" on the RS-encoded witness.

Recall AIR's polynomial constraints: $\forall C_k \in \mathcal{C}, \forall i < T,\ C_k(\mathbf{S}[i], \mathbf{S}[i+1]) = 0$.
This means $C_k(P_0(x), \ldots, P_a(x), P_0(gx), \ldots, P_a(gx)) = 0$ for all $x \in \left\{g^i \mid 0 \leq i < T\right\} = G \setminus \left\{g^T\right\}$.

Idea: Define $Z_G(X) := \prod_{h \in G}(X - h)$. Then,

$$\frac{(X - g^T)}{Z_G(X)} C_k\big((P_0(X), \ldots, P_a(X), P_0(gX), \ldots, P_a(gX)\big) \text{ must be a polynomial.}$$

We don't have access to P_j directly. But on D, it can be expressed with f_j, B_j and Z_{io}!

Instance reduction ($x_{\text{AIR}} \to x_{\text{RS-AIR}}$) [Part 2/2]

Let $\mathbf{f} = (f_0, \ldots, f_a) \in (\mathbb{F}^D)^{a+1}$. For each $C_k \in \mathcal{C}$, define $C_k[\mathbf{f}] : D \to \mathbb{F}$ s.t. for all $x \in D$:

$$C_k[\mathbf{f}](x) = \frac{(x - g^T)}{Z_G(x)} C_k\big((f_0 Z_{\text{io}} + B_0)(x), \ldots, (f_a Z_{\text{io}} + B_a)(x), (f_0 Z_{\text{io}} + B_0)(gx), \ldots, (f_a Z_{\text{io}} + B_a)(gx)\big)$$

If $\mathbf{f} = (f_0, \ldots, f_a)$ **is an encoding of a valid execution trace,**
then, for any k**,** $C_k[\mathbf{f}]$ **is a codeword of a RS code** RS**[**$\mathbb{F}, D, k_c$**].** (Here, $k_c = T + 5$)

Witness $w_{\text{RS-AIR}}$**:** an interleaved word $\mathbf{f} = (f_0, \ldots, f_a) \in (\mathbb{F}^D)^{a+1}$

Instance $x_{\text{RS-AIR}}$**:**
For input-output: $(B_j(X))_{0 \le j \le a}$ of deg < 1 and $Z_{\text{io}}(X) = (X - g^0)(X - g^T)$
Rational constraints $(C_k[\cdot])_{0 \le k \le p}$ $\quad$ $C_k[\cdot]$ and any $\mathbf{f} \in (\mathbb{F}^D)^{a+1}$ jointly define $C_k[\mathbf{f}] \in \mathbb{F}^D$
Assignment code $\text{RS}[\mathbb{F}, D, k]$ and constraint code $\text{RS}[\mathbb{F}, D, k_c]$

RS-AIR relation $\mathcal{R}_{\text{RS-AIR}}$

$$(x_{\text{RS-AIR}}, w_{\text{RS-AIR}}) \in \mathcal{R}_{\text{RS-AIR}} \iff w_{\text{RS-AIR}} = \mathbf{f} = (f_0, \ldots, f_a) \text{ satisfies } \begin{cases} \forall j, f_j \in \text{RS}[\mathbb{F}, D, k] \\ \forall k, C_k[\mathbf{f}] \in \text{RS}[\mathbb{F}, D, k_c] \end{cases}$$

Reduction: From $(x_{\text{AIR}}, w_{\text{AIR}})$, we've just defined an RS-encoded pair $(x_{\text{RS-AIR}}, w_{\text{RS-AIR}})$ satisfying:

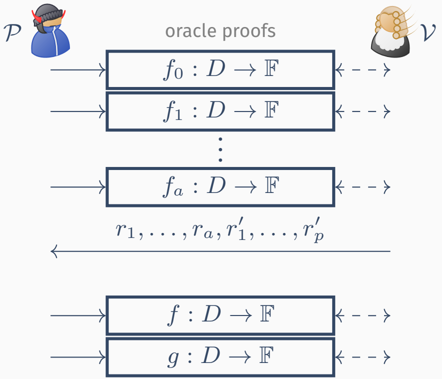
Perfect completeness If $(x_{\text{AIR}}, w_{\text{AIR}}) \in \mathcal{R}_{\text{AIR}}$, then $(x_{\text{RS-AIR}}, w_{\text{RS-AIR}}) \in \mathcal{R}_{\text{RS-AIR}}$.
Perfect soundness If $x_{\text{AIR}} \notin \mathcal{L}_{\text{AIR}}$[1], then $x_{\text{RS-AIR}} \notin \mathcal{L}_{\text{RS-AIR}}$.

[1] For a binary relation $\mathcal{R} = \{(x, w)\}$, its associated language is $\mathcal{L} = \{x \mid \exists w, (x, w) \in \mathcal{R}\}$.

Step 2: Aggregating low-degree tests via 1-round IOP

Idea: average distance to a code V

Let V be a linear code and $u_0, \ldots, u_l : D \to \mathbb{F}$. Denote Δ relative Hamming distance.
Then, $\Delta(u_0 + \sum_{j=1}^{l} r_j u_j, V) \simeq \max_j \Delta(u_j, V)$ with high proba over $r_1, \ldots, r_l$

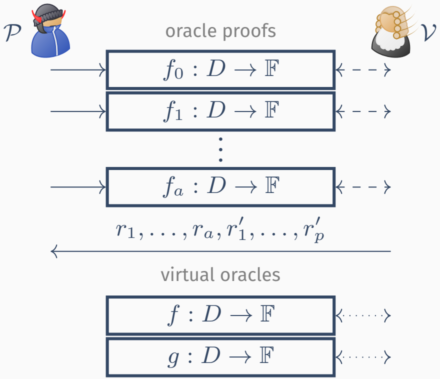


$\mathcal{P}$ computes $\begin{cases} f := f_0 + \sum_{j=1}^{a} r_j f_j \\ g := C_0[\mathbf{f}] + \sum_{k=1}^{p} r'_k C_k[\mathbf{f}] \end{cases}$

$\mathcal{P}$ and $\mathcal{V}$ check if f and g are RS codewords,
with $O(\log T)$ queries and verifier complexity.

Idea: average distance to a code V

Let V be a linear code and $u_0, \ldots, u_l : D \to \mathbb{F}$. Denote Δ relative Hamming distance.
Then, $\Delta(u_0 + \sum_{j=1}^{l} r_j u_j, V) \simeq \max_j \Delta(u_j, V)$ with high proba over $r_1, \ldots, r_l$



$\mathcal{P}$ computes $\begin{cases} f := f_0 + \sum_{j=1}^{a} r_j f_j \\ g := C_0[\mathbf{f}] + \sum_{k=1}^{p} r'_k C_k[\mathbf{f}] \end{cases}$

$\mathcal{P}$ and $\mathcal{V}$ check if f and g are RS codewords,
with $O(\log T)$ queries and verifier complexity.

Remark: $\mathcal{P}$ doesn't need to send f and g.
By querying $\mathbf{f}(x_0)$ and $\mathbf{f}(gx_0)$, $\mathcal{V}$ can compute $f(x_0)$ and each $C_k[\mathbf{f}](x_0)$, thus $g(x_0)$.

Notice that $\mathcal{V}$ computes $Z_G(x_0)$ for $x_0 \in D$ in $O(\log T)$ ops because $Z_G(X) = \prod_{h \in G}(X - h) = X^{T+1} - 1$.

Final step: Low-degree testing

If the Collatz sequence starting with $u = 42$ reaches 1 after $T = 8$ iterations,
then $f \in \mathsf{RS}[\mathbb{F}, D, k]$ and $g \in \mathsf{RS}[\mathbb{F}, D, k_c]$.

Otherwise, with very high proba,
then f is δ-far from $\mathsf{RS}[\mathbb{F}, D, k]$ or g is δ-far from $\mathsf{RS}[\mathbb{F}, D, k_c]$,
with $\delta \to 1$ when $\frac{\max(k, k_c)}{|D|} \to 0$.

An IOP with logarithmic query/verifier complexities is needed to **test proximity** to a Reed-Solomon code, meaning a verifier must distinguish between:

- functions which are RS codewords,
- functions which are δ-far from any codeword.

Univariate low-degree test: FRI protocol

Reed-Solomon Proximity Testing

Input:	a code RS$[\mathbb{F}, D, k]$, a parameter δ
Input oracle:	$f : D \to \mathbb{F}$
Completeness:	If $f \in$ RS$[\mathbb{F}, D, k]$, then $\exists P \ \Pr[\langle P, V\rangle = 1] = 1$
Soundness:	If $\Delta\left(f, \text{RS}[\mathbb{F}, D, k]\right) > \delta$, then $\forall \tilde{P} \ \Pr[\langle \tilde{P}, V\rangle = 1] < \text{err}(\delta)$ Δ relative Hamming distance

Naive test

1. Query k entries of $f \in \mathbb{F}^D$: $f(x_0), \ldots, f(x_{k-1})$,
2. Reconstruct poly P by interpolation, then evaluate it in a $(k+1)$-th point $x_k \in D$,
3. Accept iff $P(x_k) = f(x_k)$.

Soundness: $\mathcal{V}$ accepts with proba $< 1 - \delta$

Problem: # queries is **linear** in $|D|$.

$\mathcal{V}$ can't do better on his own. But a prover $\mathcal{P}$ can help.

RS IOP of Proximity

FRI Protocol

[Ben–Sasson-Bentov-Horesh-Riabzev'18]

# rounds	$< \log\|D\|$
# queries	$O(2 \log\|D\|)$
prover time	$< 6\|D\|$
verifier time	$O(21 \log\|D\|)$
oracle length	$< \|D\|/3$

Let $k = 2^r$. Assume there exists $\omega \in \mathbb{F}^\times$ of order a large power of 2, and consider evaluation domains $D := \langle \omega \rangle$ and $D' := \langle \omega^2 \rangle$ $(|D| > k)$.

How to check if $f : D \to \mathbb{F}$ is in RS$[\mathbb{F}, D, k]$?

Let $k = 2^r$. Assume there exists $\omega \in \mathbb{F}^\times$ of order a large power of 2, and consider evaluation domains $D := \langle \omega \rangle$ and $D' := \langle \omega^2 \rangle$ $(|D| > k)$.

How to check if $f : D \to \mathbb{F}$ is in RS$[\mathbb{F}, D, k]$?

Idea:

- Define $P(X)$ such that $P(x) = f(x)$ for every $x \in D$ $\qquad \deg P < |D|$
- Split P into g, h, such that $P(X) = g(X^2) + Xh(X^2)$ $\qquad \deg g, \deg h < |D|/2$
- For every $x \in D$, $f(x) = g(x^2) + x \cdot h(x^2)$
- Consider $g, h : D' \to \mathbb{F}$ with $|D'| = \frac{1}{2}|D|$
- For $\alpha \in \mathbb{F}$, define $\textsc{Fold}[f, \alpha] : D' \to \mathbb{F}$ by $\textsc{Fold}[f, \alpha](y) = g(y) + \alpha \cdot h(y)$

Halving the size of the problem by folding

Let $k = 2^r$. Assume there exists $\omega \in \mathbb{F}^\times$ of order a large power of 2, and consider evaluation domains $D := \langle \omega \rangle$ and $D' := \langle \omega^2 \rangle$ ($|D| > k$).

How to check if $f : D \to \mathbb{F}$ is in RS[$\mathbb{F}, D, k$]?

Idea:

- Define $P(X)$ such that $P(x) = f(x)$ for every $x \in D$ $\qquad \deg P < |D|$
- Split P into g, h, such that $P(X) = g(X^2) + Xh(X^2)$ $\qquad \deg g, \deg h < |D|/2$
- For every $x \in D$, $f(x) = g(x^2) + x \cdot h(x^2)$
- Consider $g, h : D' \to \mathbb{F}$ with $|D'| = \frac{1}{2}|D|$
- For $\alpha \in \mathbb{F}$, define $\textsc{Fold}[f, \alpha] : D' \to \mathbb{F}$ by $\textsc{Fold}[f, \alpha](y) = g(y) + \alpha \cdot h(y)$

$$\forall \alpha, f \in \mathrm{RS}[\mathbb{F}, D, k] \implies \textsc{Fold}[f, \alpha] \in \mathrm{RS}[\mathbb{F}, D', k/2]$$

Halving the size of the problem by folding

Let $k = 2^r$. Assume there exists $\omega \in \mathbb{F}^\times$ of order a large power of 2, and consider evaluation domains $D := \langle \omega \rangle$ and $D' := \langle \omega^2 \rangle$ $(|D| > k)$.

How to check if $f : D \to \mathbb{F}$ is in RS$[\mathbb{F}, D, k]$?

Idea:

- Define $P(X)$ such that $P(x) = f(x)$ for every $x \in D$ $\qquad \deg P < |D|$
- Split P into g, h, such that $P(X) = g(X^2) + Xh(X^2)$ $\qquad \deg g, \deg h < |D|/2$
- For every $x \in D$, $f(x) = g(x^2) + x \cdot h(x^2)$
- Consider $g, h : D' \to \mathbb{F}$ with $|D'| = \frac{1}{2}|D|$
- For $\alpha \in \mathbb{F}$, define $\textsc{Fold}[f, \alpha] : D' \to \mathbb{F}$ by $\textsc{Fold}[f, \alpha](y) = g(y) + \alpha \cdot h(y)$

$$\forall \alpha, f \in \mathrm{RS}[\mathbb{F}, D, k] \implies \textsc{Fold}[f, \alpha] \in \mathrm{RS}[\mathbb{F}, D', k/2]$$

Observe, for all $x \in D$,

$$\textsc{Fold}[f, \alpha](x^2) = \frac{f(x)+f(-x)}{2} + \alpha \frac{f(x)-f(-x)}{2x}.$$

Compute $\textsc{Fold}[f, \alpha](y)$ with only **2 queries** to f.

Folding preserves distance to the code

Notations:

- $\mathsf{RS}_0 := \mathsf{RS}[\mathbb{F}, D, k]$ and $\mathsf{RS}_1 := \mathsf{RS}[\mathbb{F}, D', k/2]$ of rate $\rho := \frac{k}{|D|}$

Let κ be a security parameter. Assume $|\mathbb{F}|$ is large enough, i.e. $O_{\rho,\delta}\left(\frac{|D|^2}{|\mathbb{F}|}\right) = \mathsf{negl}(\kappa)$.

Theorem [Ben–Sasson-Carmon-Ishai-Kopparty'20]

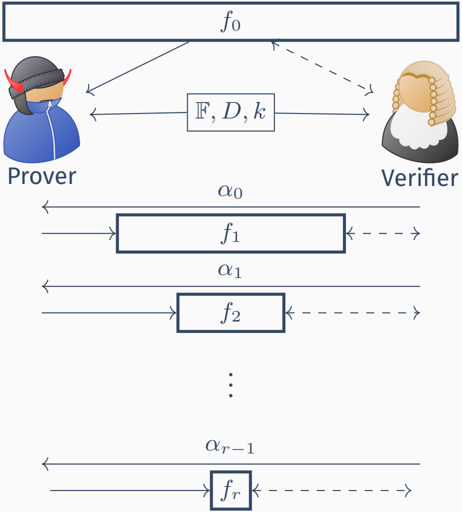
Assume $\delta < 1 - \sqrt{\rho}$. Let $g, h : D' \to \mathbb{F}$. If either $\Delta(g, \mathsf{RS}_1) > \delta$ or $\Delta(h, \mathsf{RS}_1) > \delta$, then

$$\Pr_{\alpha \in \mathbb{F}}[\Delta(g + \alpha h, \mathsf{RS}_1) < \delta] < \mathsf{negl}(\kappa)$$

Corollary

Assume $\delta < 1 - \sqrt{\rho}$. If $\Delta(f, \mathsf{RS}_0) > \delta$, then

$$\Pr_{\alpha \in \mathbb{F}}[\Delta(\textsc{Fold}[f, \alpha], \mathsf{RS}_1) < \delta] < \mathsf{negl}(\kappa)$$



Honest prover computes:

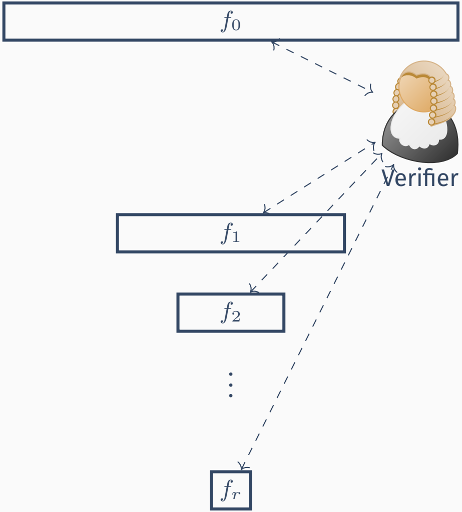
$$f_1 = \textsc{Fold}\left[f_0, \alpha_0\right]$$

$$f_2 = \textsc{Fold}\left[f_1, \alpha_1\right]$$

$$\vdots$$

$$f_r = \textsc{Fold}\left[f_{r-1}, \alpha_{r-1}\right]$$

FRI Protocol: Query Phase



Check consistency at random locations

$$f_1(s_1) \stackrel{?}{=} \textsc{Fold}\left[f_0, \alpha_0\right](s_1)$$

$$f_2(s_2) \stackrel{?}{=} \textsc{Fold}\left[f_1, \alpha_1\right](s_2)$$

$$\vdots$$

$$f_r(s_r) \stackrel{?}{=} \textsc{Fold}\left[f_{r-1}, \alpha_{r-1}\right](s_r)$$

Final test: $f_r \stackrel{?}{\in} \mathsf{RS}_r$

Soundness: If $\Delta(f, \mathsf{RS}[\mathbb{F}, D, k]) > \delta$, $\mathcal{V}$ accepts with proba $< \mathsf{err}$.

κ security parameter

Theorem

Assuming $\delta < 1 - \sqrt{\rho}$ (ρ is code rate),

$$\begin{aligned} \mathsf{err} &< \mathsf{err}_{\mathsf{commit}} + (\mathsf{err}_{\mathsf{query}})^l \\ &< \mathsf{negl}(\kappa) + (1-\delta)^l \end{aligned}$$

To get error $\mathsf{err} = \mathsf{negl}(\kappa)$, repeat query phase enough time (l times).

For instance, for $\kappa = 128$.

Take $|\mathbb{F}| > 2^{256}$, $|D| = 2^{20}$, $k = 2^{16}$, $\delta = 1 - \sqrt{\rho} - 2^{-14} \simeq 3/4$. Then, repeat $l = 65$ times the query phase.

If $\Delta(f, \mathsf{RS}[\mathbb{F}, D, k]) > \delta$, then $\mathcal{V}$ accepts with proba $< 2^{-128}$.

Beyond Reed-Solomon codes

Tensor product of RS codes

$$\mathsf{RS}[\mathbb{F}, L, d]^{\otimes m} = \left\{ f \in \mathbb{F}^{L^m} \mid P \in \mathbb{F}[X_1, \ldots, X_m], \deg_{X_i} P < d, f = P_{|L^m} \right\}$$

Reed-Muller codes

$$\mathsf{RM}[\mathbb{F}, L, d, m] = \left\{ f \in \mathbb{F}^{L^m} \mid P \in \mathbb{F}[X_1, \ldots, X_m], \deg_{\mathrm{tot}} P < d, f = P_{|L^m} \right\}$$

Is it possible to construct IOPP for RS$^{\otimes}$ and RM families with efficiency similar to the RS case?

Proximity tests for multivariate codes

Theorem (informal)

There exists an IOPP $(\mathcal{P}, \mathcal{V})$ for RS$^{\otimes}$ (resp. RM codes) with

- ✔ linear prover time
- ✔ linear (interactive) proof length
- ✔ logarithmic query complexity
- ✔ logarithmic verifier time

Theorem (informal)

There exists an IOPP $(\mathcal{P}, \mathcal{V})$ for RS$^{\otimes}$ (resp. RM codes) with

- ✓ linear prover time
- ✓ linear (interactive) proof length
- ✓ logarithmic query complexity
- ✓ logarithmic verifier time

1. Decompose m-variate polynomial f into 2^m m-variate polynomials $g_{\boldsymbol{u}}, \boldsymbol{u} \in \{0,1\}^m$.
2. Define folding of f as a random linear combination of the $g_{\boldsymbol{u}}$'s:

$$\textsc{Fold}\,[f, \boldsymbol{p}]\,(\boldsymbol{y}) = \sum_{\boldsymbol{u} \in \{0,1\}^m} \boldsymbol{p}^{\boldsymbol{u}} g_{\boldsymbol{u}}(\boldsymbol{y}). \qquad (\mathrm{RS}^{\otimes})$$

Properties:

completeness	$\textsc{Fold}\,[\cdot, \boldsymbol{p}]\,(C) \subseteq C'$
locally computable	2^m queries
distance preservation	f δ-far for $C \implies \textsc{Fold}\,[f, \boldsymbol{p}]$ δ'-far from C' (w.h.p.)

Theorem (informal)

There exists an IOPP $(\mathcal{P}, \mathcal{V})$ for RS$^{\otimes}$ (resp. RM codes) with

- ✔ linear prover time
- ✔ linear (interactive) proof length
- ✔ logarithmic query complexity
- ✔ logarithmic verifier time

1. Decompose m-variate polynomial f into 2^m m-variate polynomials $g_{\boldsymbol{u}}, \boldsymbol{u} \in \{0,1\}^m$.
2. Define folding of f as a random linear combination of the $g_{\boldsymbol{u}}$'s:

$$\textsc{Fold}\,[f, (\boldsymbol{p}, \boldsymbol{q})]\,(\boldsymbol{y}) = \sum_{\boldsymbol{u} \in \{0,1\}^m} \boldsymbol{p}^{\boldsymbol{u}} g_{\boldsymbol{u}}(\boldsymbol{y}) + \sum_{\boldsymbol{u} \in \{0,1\}^m \setminus \{\boldsymbol{0}\}} \boldsymbol{q}^{\boldsymbol{u}} \boldsymbol{y}^{\boldsymbol{u}} g_{\boldsymbol{u}}(\boldsymbol{y}). \tag{RM}$$

Properties:

completeness	$\textsc{Fold}\,[\cdot, \boldsymbol{p}]\,(C) \subseteq C'$
locally computable	2^m queries
distance preservation	f δ-far for $C \implies \textsc{Fold}\,[f, \boldsymbol{p}]$ δ'-far from C' (w.h.p.)

Theorem (informal)

There exists an IOPP $(\mathcal{P}, \mathcal{V})$ for RS$^{\otimes}$ (resp. RM codes) with

- ✔ linear prover time
- ✔ linear (interactive) proof length
- ✔ logarithmic query complexity
- ✔ logarithmic verifier time

Soundness (informal)

Let δ be the relative distance of f to RM (resp. RS$^{\otimes}$).

Assuming $\delta < c$, $\mathsf{err}(\delta) < \mathsf{negl} + (1-\delta)^l$.

✖ Soundness threshold c not as good as $1 - \sqrt{\rho}$ (RS case).
$\rightarrow$ greater repetition parameter l, i.e. more queries, thus longer non-interactive proofs.

The rest of the story.

Recall: arithmetization transforms “instructions set” of a program into constraints on low-degree polynomials, e.g. vanish on a given set.

We can use our IOPP to check that $S(\boldsymbol{X})$ committed via $S_{|H^m} : H^m \to \mathbb{F}$ vanishes on a set G^m, where $G \cap H = \emptyset$ with a succinct proof.

What about other codes?

With Jade Nardi: a “FRI-like” IOPP for some families of **Algebraic Geometry codes**.

`https://eccc.weizmann.ac.il/report/2020/165/`

The rest of the story.

Recall: arithmetization transforms “instructions set” of a program into constraints on low-degree polynomials, e.g. vanish on a given set.

We can use our IOPP to check that $S(\boldsymbol{X})$ committed via $S_{|H^m} : H^m \to \mathbb{F}$ vanishes on a set G^m, where $G \cap H = \emptyset$ with a succinct proof.

What about other codes?

With Jade Nardi: a “FRI-like” IOPP for some families of **Algebraic Geometry codes**.

`https://eccc.weizmann.ac.il/report/2020/165/`

Future work:

- Construct building-blocks for multivariate/AG code-based arithmetization
- Find more “nice families” of AG codes
- Improve soundness of AG-IOPP

Thank you for your attention!