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.

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

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 $\pi$

Weak Verifier
On input $(F, x, y, \pi)$, accept iff $\pi$ 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
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 $\pi$

**Weak Verifier**

On input $(F, x, y, \pi)$, accept iff $\pi$ 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)

<table>
<thead>
<tr>
<th>Year</th>
<th>CRHF</th>
<th>DLOG</th>
<th>KoE/AGM/GGM (pairing-based)</th>
<th>Group of unknown order</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td></td>
<td></td>
<td>Pinocchio [PGHR]</td>
<td></td>
</tr>
<tr>
<td>2014</td>
<td></td>
<td></td>
<td>[BCTV]</td>
<td></td>
</tr>
<tr>
<td>2016</td>
<td>ZKBoo [GMO16]</td>
<td>[BCCGP16]</td>
<td>[Groth16]</td>
<td>[GM17]</td>
</tr>
<tr>
<td></td>
<td>SCI [BBC+]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2017</td>
<td>Ligero [AHIV]</td>
<td>Bulletproof [BBB+]</td>
<td>(ZK) vSQL [ZGK+]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Hyrax [WTS+]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td>Stark [BBHR]</td>
<td></td>
<td>vRAM [ZGK+]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Aurora [BCR+]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RedShift [KPV]</td>
<td></td>
<td>Marlin [CHM+]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Virgo [ZXZS]</td>
<td></td>
<td>Libra [XZZ+]</td>
<td></td>
</tr>
<tr>
<td>2020</td>
<td>Virgo++ [ZWZZ]</td>
<td></td>
<td>Mirage [KKPS]</td>
<td></td>
</tr>
</tbody>
</table>

Some implementations of succinct non-interactive arguments for general computations
PCP-based succinct non-interactive arguments
Starting point: PCP characterization of NP

**PCP class**

$L \in \text{PCP}[r, q]$ if \( \exists \) efficient randomized \( V \) such that

- **Completeness:** \( \forall x \in L, \exists \pi, V^\pi(x) = 1 \)
- **Soundness:** \( \forall x \notin L, \forall \tilde{\pi}, \Pr[V^{\tilde{\pi}}(x) = 0] > 1/2 \)

where \( V \) reads \( \pi \) at \( \leq q \) locations and tosses \( \leq r \) coins.

**NP class**

$L \in \text{NP}$ if \( \exists \) efficient \( V \) such that:

- **Completeness:** \( \forall x \in L, \exists w, V(x, w) = 1 \)
- **Soundness:** \( \forall x \notin L, \forall \tilde{w}, V(x, \tilde{w}) = 0 \)

**PCP Theorem:** \( \text{NP} = \text{PCP}[\log n, O(1)] \)

- BFLS: nine.osf/nine.osf
- FGL: plus.osf/nine.osf/six.osf
- ALMSS': nine.osf/eight.osf
- AS': nine.osf/eight.osf

PCPs are not succinct proofs! PCP generation is too expensive! Three.osf years later: practical real-world deployment.
Starting point: PCP characterization of NP

**PCP class**

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

**Completeness:** \( \forall x \in L, \exists \pi, V^\pi(x) = 1 \)

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

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

**NP class**

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

**Completeness:** \( \forall x \in L, \exists w, V(x, w) = 1 \)

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

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

✓ Check NP statements way faster than checking an NP witness!

❌ PCPs are not succinct proofs! ❌ PCP generation is too expensive!

ynamo: 30 years later: practical real-world deployment
Interactive Proof Systems and Zero-Knowledge

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

**IP class**

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

- **Completeness:** $\forall x \in L, \exists P, \langle P, V \rangle(x) = 1$
- **Soundness:** $\forall x \notin L, \forall \bar{P}, \Pr[\langle \bar{P}, V \rangle(x) = 0] > \frac{1}{2}$

🔍 **Thm:** $\text{IP} = \text{PSPACE}$ [Shamir’86]

Interactive Proofs (IPs) can be

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

🔍 Assuming the existence of one-way functions, all languages in $\text{NP}$ have a ZK proof system. [Goldreich-Micali-Wigderson’91]

- **Public-coin:** $V$ 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 from PCPs

Succinct interactive arguments for $\textbf{NP}$

[Kilian'92]

Computationally bounded

Prover

Verifier

Collision-resistant hash function $H$ (CRHF)

$\text{Commit}(\pi)$

$\text{MerkleRoot}(\pi)$

random queries $q_1, q_2, q_3$

$\text{Reveal}(\pi[q_1]), \text{Reveal}(\pi[q_2]), \text{Reveal}(\pi[q_3])$

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

Check Merkle paths and run $\nu_{\text{PCP}}$

[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

[Prover] Computationally bounded

[Verifier]

PCP \( \pi \) for \( x \in L \)

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

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

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

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

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

\( \pi = (h_0, \pi[q_1], \pi[q_2], \pi[q_3], p_1, p_2, p_3) \)

\( \pi = (h_0, \pi[q_1], \pi[q_2], \pi[q_3], p_1, p_2, p_3) \)

Derive queries,
check Merkle paths
and run \( \nu_{PCP} \)

- 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

[Prover] Computationally bounded → Same public-randomness i.e. a CRHF $H$ → [Verifier]

PCP $\pi$ for $x \in L$

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

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

$p_1 = \text{MerklePath}(\pi[q_1])$
$p_2 = \text{MerklePath}(\pi[q_2])$
$p_3 = \text{MerklePath}(\pi[q_3])$

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

Derive queries, check Merkle paths and run $V_{PCP}$

✓ Succinct argument
✓ One message
✓ Presumably post-quantum
✓ Lightweight crypto
✗ PCP generation is too expensive

✓ Non-interactive in the Random Oracle model (→ SNARG)
✓ Compatible with: Zero-Knowledge, Proof of Knowledge (→ ZK-SNARK)
Succinct Non-interactive ARGuments from PCPs

**Applying Fiat-Shamir Paradigm to PCPs**

- **Prover**
  - Computationally bounded
  - PCP $\pi$ for $x \in L$
    - $h_0 = \text{MerkleRoot}(\pi)$
  - Derive queries $q_1, q_2, q_3$ from $H(h_0)$
    - $p_1 = \text{MerklePath}(\pi[q_1])$
    - $p_2 = \text{MerklePath}(\pi[q_2])$
    - $p_3 = \text{MerklePath}(\pi[q_3])$

- **Verifier**
  - Same public-randomness
    - i.e. a CRHF $H$

\[ \pi = (h_0, \pi[q_1], \pi[q_2], \pi[q_3], p_1, p_2, p_3) \]

Derive queries, check Merkle paths and run $V_{PCP}$

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

- **✓ Non-interactive in the Random Oracle model (→ SNARG)**
- **✓ Compatible with: Zero-Knowledge, Proof of Knowledge (→ ZK-SNARK)**
Succinct Non-interactive ARGuments from PCPs

PCP \( \pi \) for \( x \in L \)

\[ h_0 = \text{MerkleRoot}(\pi) \]

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

\[ p_1 = \text{MerklePath}(\pi[q_1]) \]
\[ p_2 = \text{MerklePath}(\pi[q_2]) \]
\[ p_3 = \text{MerklePath}(\pi[q_3]) \]

\[ \pi = (h_0, \pi[q_1], \pi[q_2], \pi[q_3], p_1, p_2, p_3) \]

Derive queries, check Merkle paths and run \( \nu_{PCP} \)

✅ 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 (Interactive Oracle Proofs)

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: PCP + hash function = succinct arguments

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

[BCS16, RRR’16]
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 Transparent ARgument of Knowledge [BBHR18]

non-interactive argument for bounded halting problems of a Random-Access Machine (RAM)

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

<table>
<thead>
<tr>
<th>Setup</th>
<th>Prover</th>
<th>Verifier</th>
<th>Communication complexity</th>
<th>Post-Quantum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transparent</td>
<td>(O(T \log^2 T))</td>
<td>(O(\log^2 T))</td>
<td>(O(\log^2 T))</td>
<td>yes</td>
</tr>
</tbody>
</table>

**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 $(P, V)$ with a polylog verifier, logarithmic query complexity, linear oracle proof length, quasilinear prover.
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.”
The initial relation to build a STARK

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

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

**AIR relation** \(R_{\text{AIR}}\)
\[
(x_{\text{AIR}}, w_{\text{AIR}}) \in R_{\text{AIR}} \iff \begin{cases} \text{"input is } u \" \\ \text{"output is } z\" \\ \forall C \in C, \forall i < T, C(S[i], S[i + 1]) = 0 \end{cases}
\]
An AIR for our toy example

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

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

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

<table>
<thead>
<tr>
<th>$i$</th>
<th>$R_0$</th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$R_3$</th>
<th>$R_4$</th>
<th>$R_5$</th>
<th>$R_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Instance $x_{\text{AIR}}$:  

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

Constraints $C = \{C_0, \ldots, C_7\} \subset \mathbb{F}[X, Y]$
3. For $j = 0, \ldots, 6$, $C_j(X, Y) = X_j^2 - X_j$
4. $C_7(X, Y) = (1 - X_0) (\langle b, X \rangle - 2\langle b, Y \rangle) + X_0 (3\langle b, X \rangle + 1 - \langle b, Y \rangle)$

$(x_{\text{AIR}}, w_{\text{AIR}}) \in R_{\text{AIR}} \iff \begin{cases} \langle b, S[0] \rangle - 42 = 0 \\
\langle b, S[T] \rangle - 1 = 0 \\
\forall C_k \in C, \forall i < T, C_k(S[i], S[i+1]) = 0 \end{cases}$
In the previous episode

- Constraints of a given computation captured by relation $R$
  - Constraints on low-degree polynomials (e.g. vanish on a given set)
  - Low-degree testing

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

Proof system $(P, V)$

Program $\rightarrow$ front-end compiler $\rightarrow$ Relation $R$ (Intermediate representation) $\rightarrow$ Arithmetization $\rightarrow$ Proximity to a code $C$
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 \( \rho | D | = T \).

Reed-Solomon code of dim. \( k \) : \( \text{RS}[\mathbb{F}, D, k] := \{ P | D : D \rightarrow \mathbb{F} \mid P \in \mathbb{F}[X], \deg P < k \} \).

**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 \( D \).

\[
\begin{array}{cccccccc}
  & R_0 & R_1 & R_2 & R_3 & R_4 & R_5 & R_6 \\
 i = 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
 i = 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
 i = 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 i = 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
 i = 4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 i = 5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
 i = 6 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 i = 7 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
 i = 8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
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 \( \rho |D| = T \).

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

**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 \( D \).

<table>
<thead>
<tr>
<th>( g^0 )</th>
<th>( g^1 )</th>
<th>( g^2 )</th>
<th>( g^3 )</th>
<th>( g^4 )</th>
<th>( g^5 )</th>
<th>( g^6 )</th>
<th>( g^7 )</th>
<th>( g^8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\( \rho \in (0, 1) \)
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 \( \rho \mid D \mid = T \).

Reed-Solomon code of dim. \( k \) : \( \text{RS}[\mathbb{F}, D, k] := \{ P_D : D \to \mathbb{F} \mid P \in \mathbb{F}[X], \deg P < k \} \).

**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 \( D \):

\[
\begin{align*}
\begin{array}{cccccccc}
\hline
P_0 & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 \\
\hline
P_0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
P_1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
P_2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
P_3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
P_4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
P_5 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
P_6 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
P_7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\end{align*}
\]

\( \rho \in (0, 1) \)
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** \((x_{\text{AIR}} \rightarrow x_{\text{RS-AIR}}) [\text{Part 1/2}]\)

Define \((a + 1)\) “boundary” polynomials \((B_j(X))_{0 \leq j \leq a}\) such that \(\deg B_j < 2\),

\[
\begin{align*}
(B_j(g^0))_{0 \leq j \leq a} &= (0, 1, 0, 1, 0, 1, 0) \\
B_j(g^T) &= \begin{cases} 
1 & \text{if } j = 0, \\
0 & \text{otherwise.}
\end{cases}
\]

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

**Witness reduction** \((w_{\text{AIR}} \rightarrow w_{\text{RS-AIR}})\) 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 \(\frac{P_j(X) - B_j(X)}{Z_{\text{io}}(X)}\) on \(D\) to get \(f_j : D \rightarrow \mathbb{F}\) (expected to be a poly of \(\deg \leq T - 2\))

\[
\begin{align*}
\{ & P_j(g^0) = B_j(g^0) \} & \iff & \{ & (X - 1) | (P_j(X) - B_j(X)) \\
\{ & P_j(g^T) = B_j(g^T) \} & & \{ & (X - g^T) | (P_j(X) - B_j(X)) \\
\end{align*}
\]
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 \((B_j(X))_{0 \leq j \leq a}\) such that \(\deg B_j < 2\),

\[
\left( B_j(g^0) \right)_{0 \leq j \leq a} = (0, 1, 0, 1, 0, 1, 0)
\]

\(\text{input} = 42\)

and

\[
B_j(g^T) = \begin{cases} 1 & \text{if } j = 0, \\ 0 & \text{otherwise}. \end{cases}
\]

\(\text{output} = 1\)

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 \(\leq T\) 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 \(\deg \leq T - 2\))

If \(f = (f_0, \ldots, f_a)\) is an encoding of a valid execution trace, then, for any \(j\), \(f_j\) is a codeword of a code \(\text{RS}[\mathbb{F}, D, k]\). (Here, \(k = T - 1\))
Step 1: Rational functions which are low-degree polynomials

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

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

Recall AIR’s polynomial constraints: $\forall C_k \in C, \forall i < T, C_k(S[i], S[i+1]) = 0$.
This means $C_k(P_0(x), \ldots, P_a(x), P_0(gx), \ldots, P_a(gx)) = 0$ for all $x \in \{g^i \mid 0 \leq i < T\} = G \setminus \{g^T\}$.
Step 1: Rational functions which are low-degree polynomials

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

**Second**, define “rational constraints” on the RS-encoded witness.

Recall AIR’s polynomial constraints: \( \forall C_k \in C, \forall i < T, C_k(S[i], S[i + 1]) = 0 \).

This means \( C_k(P_0(x), \ldots, P_a(x), P_0(gx), \ldots, P_a(gx)) = 0 \) for all \( x \in \{g^i | 0 \leq i < T\} = G \setminus \{g^T\} \).

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

\[
\frac{(X - g^T)}{Z_G(X)} C_k \left( (P_0(X), \ldots, P_a(X), P_0(gX), \ldots, P_a(gX)) \right)
\]

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: $\forall C_k \in C, \forall i < T, C_k(S[i], S[i + 1]) = 0$.
This means $C_k(P_0(x), \ldots, P_a(x), P_0(gx), \ldots, P_a(gx)) = 0$ for all $x \in \{g^i | 0 \leq i < T\} = G \setminus \{g^T\}$.

Idea: Define $Z_G(X) := \prod_{h \in G}(X - h)$. Then,
\[
\frac{x - g^T}{Z_G(x)} C_k\left(\left(P_0(X), \ldots, P_a(X), P_0(gX), \ldots, P_a(gX)\right)\right)
\] 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 $Z_{\text{io}}$!

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

Let $f = (f_0, \ldots, f_a) \in (\mathbb{F}^D)^{a+1}$. For each $C_k \in C$, define $C_k[f] : D \rightarrow \mathbb{F}$ s.t. for all $x \in D$:
\[
C_k[f](x) = \frac{x - g^T}{Z_G(x)} C_k\left(\left(f_0Z_{\text{io}} + B_0\right)(x), \ldots, \left(f_aZ_{\text{io}} + B_a\right)(x), \left(f_0Z_{\text{io}} + B_0\right)(gx), \ldots, \left(f_aZ_{\text{io}} + B_a\right)(gx)\right)
\]
Step 1: Rational functions which are low-degree polynomials

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

**Second**, define “rational constraints” on the RS-encoded witness.

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

**Idea:** Define $Z_G(X) := \prod_{h \in G}(X - h)$. Then,
$$\frac{(X - g^T)}{Z_G(X)} C_k((P_0(X), \ldots, P_a(X), P_0(gX), \ldots, P_a(gX))$$
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** $(x_{AIR} \rightarrow x_{RS-AIR})$ [Part 2/2]

Let $f = (f_0, \ldots, f_a) \in (\mathbb{F}^D)^{a+1}$. For each $C_k \in \mathcal{C}$, define $C_k[f] : D \rightarrow \mathbb{F}$ s.t. for all $x \in D$:
$$C_k[f](x) = \frac{(x - g^T)}{Z_G(x)} C_k((f_0Z_{io} + B_0)(x), \ldots, (f_aZ_{io} + B_a)(x), (f_0Z_{io} + B_0)(gx), \ldots, (f_aZ_{io} + B_a)(gx))$$

If $f = (f_0, \ldots, f_a)$ is an encoding of a valid execution trace, then, for any $k$, $C_k[f]$ is a codeword of a RS code $RS[\mathbb{F}, D, k_c]$.  
(Here, $k_c = T + 5$)
Witness $w_{RS-AIR}$: an interleaved word $f = (f_0, \ldots, f_a) \in (\mathbb{F}^D)^{a+1}$

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

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

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

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

Perfect completeness
If $(x_{AIR}, w_{AIR}) \in \mathcal{R}_{AIR}$, then $(x_{RS-AIR}, w_{RS-AIR}) \in \mathcal{R}_{RS-AIR}$.

Perfect soundness
If $x_{AIR} \notin \mathcal{L}_{AIR}$, then $x_{RS-AIR} \notin \mathcal{L}_{RS-AIR}$.

1For 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 $\Delta$ 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

\[
\begin{align*}
\mathcal{P} &\text{ computes } \left\{ \begin{array}{l}
    f := f_0 + \sum_{j=1}^a r_j f_j \\
    g := C_0[f] + \sum_{k=1}^p r'_k C_k[f]
\end{array} \right.
\end{align*}
\]

$\mathcal{P}$ and $\mathcal{V}$ check if $f$ and $g$ are RS codewords, with $O(\log T)$ queries and verifier complexity.
Step 2: Aggregating low-degree tests via 1-round IOP

Idea: average distance to a code $V$

Let $V$ be a linear code and $u_0, \ldots, u_l : D \rightarrow \mathbb{F}$. Denote $\Delta$ 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 $r_1, \ldots, r_l$.

\[ P \text{ computes } \left\{ \begin{array}{l}
f := f_0 + \sum_{j=1}^{a} r_j f_j \\
g := C_0[f] + \sum_{k=1}^{p} r'_k C_k[f]
\end{array} \right. \]

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

Remark: $P$ doesn’t need to send $f$ and $g$.

By querying $f(x_0)$ and $f(gx_0)$, $V$ can compute $f(x_0)$ and each $C_k[f](x_0)$, thus $g(x_0)$.

Notice that $V$ 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 $f \in \text{RS}[F, D, k]$ and $g \in \text{RS}[F, D, k_c]$.

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

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 $\delta$-far from any codeword.
FRI: Prover-efficient RS IOPP (IOP of Proximity)

Reed-Solomon Proximity Testing

**Input:** a code RS[F, D, k], a parameter δ

**Input oracle:** \( f : D \rightarrow F \)

**Completeness:** If \( f \in RS[F, D, k] \), then \( \exists P \ Pr[\langle P, V \rangle = 1] = 1 \)

**Soundness:** If \( \Delta (f, RS[F, D, k]) > \delta \), then \( \forall \tilde{P} \ Pr[\langle \tilde{P}, V \rangle = 1] < \text{err}(\delta) \)

\( \Delta \) relative Hamming distance

**Naive test**

1. Query \( k \) entries of \( f \in F^D : f(x_0), \ldots, f(x_{k-1}) \),
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 | \(< |D|/3 \) |
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 $\text{RS}[\mathbb{F}, D, k]$?
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 two, 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 \( \text{RS}[\mathbb{F}, D, k] \)?

**Idea:**

- Define \( P(X) \) such that \( P(x) = f(x) \) for every \( x \in D \) \( \quad \deg P < |D| \)
- Split \( P \) into \( g, h \), such that \( P(X) = g(X^2) + Xh(X^2) \) \( \quad \deg g, \deg h < |D|/2 \)
- 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 \( \text{FOLD}[f, \alpha] : D' \to \mathbb{F} \) by \( \text{FOLD}[f, \alpha](y) = g(y) + \alpha \cdot h(y) \)
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 $\text{RS}[\mathbb{F}, D, k]$?

Idea:

- Define $P(X)$ such that $P(x) = f(x)$ for every $x \in D$ \quad $\deg P < |D|$
- Split $P$ into $g, h$, such that $P(X) = g(X^2) + Xh(X^2)$ \quad $\deg g, \deg h < |D| / 2$
- 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 $\text{FOLD}[f, \alpha] : D' \to \mathbb{F}$ by $\text{FOLD}[f, \alpha](y) = g(y) + \alpha \cdot h(y)$

\[
\forall \alpha, f \in \text{RS}[\mathbb{F}, D, k] \implies \text{FOLD}[f, \alpha] \in \text{RS}[\mathbb{F}, D', k/2]
\]
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 $\text{RS}[^F, D, k]$?

**Idea:**

- Define $P(X)$ such that $P(x) = f(x)$ for every $x \in D$ \hspace{1cm} $\deg P < |D|$
- Split $P$ into $g, h$, such that $P(X) = g(X^2) + Xh(X^2)$ \hspace{1cm} $\deg g, \deg h < |D|/2$
- 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 $\text{FOLD} [f, \alpha] : D' \to \mathbb{F}$ by $\text{FOLD} [f, \alpha] (y) = g(y) + \alpha \cdot h(y)$

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

Observe, for all $x \in D$,

$$\text{FOLD} [f, \alpha] \left( x^2 \right) = \frac{f(x) + f(-x)}{2} + \alpha \frac{f(x) - f(-x)}{2x}.$$  

Compute $\text{FOLD} [f, \alpha] (y)$ with only 2 queries to $f$.  


Folding preserves distance to the code

Notations:

- $RS_0 := RS[F, D, k]$ and $RS_1 := RS[F, D', k/2]$ of rate $\rho := k/|D|$

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

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

Assume $\delta < 1 - \sqrt{\rho}$. Let $g, h : D' \to F$. If either $\Delta(g, RS_1) > \delta$ or $\Delta(h, RS_1) > \delta$, then

$$\Pr_{\alpha \in F} \left[ \Delta(g + \alpha h, RS_1) < \delta \right] < \text{negl}(\kappa)$$

**Corollary**

Assume $\delta < 1 - \sqrt{\rho}$. If $\Delta(f, RS_0) > \delta$, then

$$\Pr_{\alpha \in F} \left[ \Delta(\text{FOLD}[f, \alpha], RS_1) < \delta \right] < \text{negl}(\kappa)$$
FRI Protocol: Commit Phase

Honest prover computes:

\[ f_1 = \text{FOLD} [f_0, \alpha_0] \]
\[ f_2 = \text{FOLD} [f_1, \alpha_1] \]
\[ \vdots \]
\[ f_r = \text{FOLD} [f_{r-1}, \alpha_{r-1}] \]
FRI Protocol: Query Phase

Check consistency at random locations

\[ f_1(s_1) = \text{FOLD}[f_0, \alpha_0](s_1) \]
\[ f_2(s_2) = \text{FOLD}[f_1, \alpha_1](s_2) \]
\[ \vdots \]
\[ f_r(s_r) = \text{FOLD}[f_{r-1}, \alpha_{r-1}](s_r) \]

Final test: \( f_r \in RS_r \)
Soundness of the FRI Protocol

**Soundness:** If $\Delta(f, RS[F, D, k]) > \delta$, $\nu$ accepts with proba $< \text{err}$.

$\kappa$ security parameter

**Theorem**

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

$$\text{err} < \text{err}_{\text{commit}} + (\text{err}_{\text{query}})^l$$

$$< \text{negl}(\kappa) + (1 - \delta)^l$$

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

For instance, for $\kappa = 128$.

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

If $\Delta(f, RS[F, D, k]) > \delta$, then $\nu$ accepts with proba $< 2^{-128}$. 
Beyond Reed-Solomon codes
Tensored RS codes, Reed-Muller codes

Tensor product of RS codes

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

Reed-Muller codes

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

Is it possible to construct IOPP for RS\(^\otimes\) and RM families with efficiency similar to the RS case?
Theorem (informal)

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

- ✔ linear prover time
- ✔ linear (interactive) proof length
- ✔ logarithmic query complexity
- ✔ logarithmic verifier time
Proximity tests for multivariate codes

**Theorem (informal)**

There exists an IOPP \((P, V)\) 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 \(g_u, u \in \{0, 1\}^m\).
2. Define folding of \(f\) as a random linear combination of the \(g_u\)’s:

\[
FOLD [f, p] (y) = \sum_{u \in \{0, 1\}^m} p^u g_u (y). 
\]  

\((RS^{\otimes})\)

**Properties:**

- Completeness: \(FOLD [\cdot, p] (C) \subseteq C'\)
- Locally computable: \(2^m\) queries
- Distance preservation: \(f \delta\)-far for \(C\) \(\implies\) \(FOLD [f, p] \delta'\)-far from \(C'\) (w.h.p.)
Proximity tests for multivariate codes

Theorem (informal)
There exists an IOPP \((\mathcal{P}, \mathcal{V})\) for \(\text{RS}\otimes\) (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 \(g_u, u \in \{0, 1\}^m\).

2. Define folding of \(f\) as a random linear combination of the \(g_u\)'s:

\[
\text{FOLD} [f, (p, q)] (y) = \sum_{u \in \{0,1\}^m} p^u g_u(y) + \sum_{u \in \{0,1\}^m\setminus\{0\}} q^u y^u g_u(y). \quad \text{(RM)}
\]

Properties:
- completeness \(\text{FOLD} [\cdot, p] (C) \subseteq C'\)
- locally computable \(2^m\) queries
- distance preservation \(f \delta\)-far for \(C \implies \text{FOLD} [f, p] \delta'\)-far from \(C'\) (w.h.p.)
**Theorem (informal)**
There exists an IOPP \((P, V)\) for \(\text{RS} \otimes\) (resp. \(\text{RM}\) codes) with
- ✓ linear prover time
- ✓ linear (interactive) proof length
- ✓ logarithmic query complexity
- ✓ logarithmic verifier time

**Soundness (informal)**
Let \(\delta\) be the relative distance of \(f\) to \(\text{RM}\) (resp. \(\text{RS} \otimes\)).

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

✗ Soundness threshold \(c\) not as good as \(1 - \sqrt{\rho}\) (\(\text{RS}\) case).
→ greater repetition parameter \(l\), i.e. more queries, thus longer non-interactive proofs.
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 \( S(X) \) 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/
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 $S(X)$ committed via $S_{|H^m} : H^m \rightarrow \mathbb{F}$ 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!