Efficient multivariate low-degree tests
via interactive oracle proofs of proximity

Sarah Bordage
Ecole Polytechnique, Institut Polytechnique de Paris & Inria
Joint work with Daniel Augot and Jade Nardi

WCC 2022
March 8, 2022
Proximity testing for a code

Given a (linear) code $C \subset \mathbb{F}^n$ and oracle access to $f \in \mathbb{F}^n$, algorithmically distinguish between

$$\begin{cases} f \in C \\ f \text{ is } \delta\text{-far from } C \end{cases}$$

with $q = o(n)$ queries to $f$.

Given domain $L$ of size $n$, identify $\mathbb{F}^n$ with $\mathbb{F}^L$: **codewords** of linear code $C \subset \mathbb{F}^n \leftrightarrow \text{functions}$ in $\mathbb{F}^L$. 
Topic of this talk

**Proximity testing for a code**

Given a (linear) code $C \subset \mathbb{F}^n$ and oracle access to $f \in \mathbb{F}^n$, algorithmically distinguish between

\[
\begin{cases}
    f \in C \\
    f \text{ is } \delta\text{-far from } C
\end{cases}
\]

with $q = o(n)$ queries to $f$.

Given domain $L$ of size $n$, identify $\mathbb{F}^n$ with $\mathbb{F}^L$: **codewords** of linear code $C \subset \mathbb{F}^n \leftrightarrow \text{functions}$ in $\mathbb{F}^L$.

**Multivariate polynomial codes**

Assume $L \subset \mathbb{F}$ and $d < |L|$. 

Topic of this talk

**Proximity testing for a code**

Given a (linear) code $C \subset \mathbb{F}^n$ and oracle access to $f \in \mathbb{F}^n$, algorithmically distinguish between

\[
\begin{aligned}
&\begin{cases}
  f \in C \\
  f \text{ is } \delta\text{-far from } C
\end{cases}
\end{aligned}
\text{ with } q = o(n) \text{ queries to } f.
\]

Given domain $L$ of size $n$, identify $\mathbb{F}^n$ with $\mathbb{F}^L$: codewords of linear code $C \subset \mathbb{F}^n \leftrightarrow \text{functions}$ in $\mathbb{F}^L$.

**Multivariate polynomial codes**

Assume $L \subset \mathbb{F}$ and $d < |L|$.

- **Tensor product of RS codes:**
  
  \[
  \text{RS}[L, d]^\otimes m = \{ f : L^m \to \mathbb{F} \mid \text{f evaluation of a poly in } \mathbb{F}[X_1, \ldots, X_m] \text{ with individual degrees } < d \}\]
Topic of this talk

Proximity testing for a code

Given a (linear) code $C \subset \mathbb{F}^n$ and oracle access to $f \in \mathbb{F}^n$, algorithmically distinguish between
\[
\begin{cases}
  f \in C \\
  f \text{ is } \delta\text{-far from } C
\end{cases}
\]
with $q = o(n)$ queries to $f$.

Given domain $L$ of size $n$, identify $\mathbb{F}^n$ with $\mathbb{F}^L$: codewords of linear code $C \subset \mathbb{F}^n \leftrightarrow$ functions in $\mathbb{F}^L$.

Multivariate polynomial codes

Assume $L \subset \mathbb{F}$ and $d < |L|$.

- **Tensor product of RS codes:**
  \[
  \text{RS}[L, d] \otimes_m = \{ f : L^m \to \mathbb{F} \mid \text{f evaluation of a poly in } \mathbb{F}[X_1, \ldots, X_m] \text{ with individual degrees } < d \}
  \]

- **Reed-Muller codes:**
  \[
  \text{RM}[L, d, m] = \{ f : L^m \to \mathbb{F} \mid \text{f evaluation of a poly in } \mathbb{F}[X_1, \ldots, X_m] \text{ of total degree } < d \}
  \]
**Proximity testing for a code**

Given a (linear) code $C \subset \mathbb{F}^n$ and oracle access to $f \in \mathbb{F}^n$, algorithmically distinguish between

$$\begin{cases} f \in C \\
 f \text{ is } \delta\text{-far from } C
\end{cases}$$

with $q = o(n)$ queries to $f$.

**Multivariate polynomial codes**

Assume $L \subset \mathbb{F}$ and $d < |L|$.

- **Tensor product of RS codes:**
  $$\text{RS}[L, d]^\otimes m = \{ f : L^m \rightarrow \mathbb{F} \mid f \text{ evaluation of a poly in } \mathbb{F}[X_1, \ldots, X_m] \text{ with individual degrees } < d \}$$

- **Reed-Muller codes:**
  $$\text{RM}[L, d, m] = \{ f : L^m \rightarrow \mathbb{F} \mid f \text{ evaluation of a poly in } \mathbb{F}[X_1, \ldots, X_m] \text{ of total degree } < d \}$$

Proximity tests for polynomial codes $\leftrightarrow$ **Low-degree tests**
Multivariate polynomial codes are locally testable

Axis-parallel tests
→ Test individual degree

Random line tests
→ Test total degree
(Require evaluation domain = \( \mathbb{F}^m \))
Multivariate polynomial codes are locally testable

- **Axis-parallel tests**
  - Test individual degree

- **Random line tests**
  - Test total degree
  (Require evaluation domain = $\mathbb{F}^m$)

- **Oracle access to $f$ only**
  - At least $d$ queries
Testing proximity in the PCP model

- Ask a prover to provide an **auxiliary proof** $\pi$
Testing proximity in the PCP model

Probabilistically Checkable Proof of Proximity (PCPP):

- Ask a prover to provide an auxiliary proof $\pi$
- Enable sublinear tests for non-locally testable codes (e.g. Reed-Solomon codes)
Testing proximity in the PCP model

Probabilistically Checkable Proof of Proximity (PCPP):

- Ask a prover to provide an **auxiliary proof** $\pi$
- Enable sublinear tests for **non-locally testable codes** (e.g. Reed-Solomon codes)
- Prover must compute in advance the answers to all possible queries $\rightarrow$ **impractical**
An Interactive Oracle Proof of Proximity (IOPP) \((\mathcal{P}, \mathcal{V})\) for \(C\) with soundness error \(s : (0, 1] \rightarrow [0, 1]\) satisfies:

**Completeness**
If \(f \in C\), then \(\exists \mathcal{P} \Pr[\langle \mathcal{P}(f), \mathcal{V}^f \rangle = 1] = 1\).

**Soundness**
If \(f\) is \(\delta\)-far from \(C\), Then, for all unbounded \(\mathcal{P}', \Pr[\langle \mathcal{P}', \mathcal{V}^f \rangle = 1] \leq s(\delta)\).
Motivation: (ZK-)SNARKs

SNARKs are “succinct proofs” that enable to check correctness of a computation, with/without ZK.
Motivation: (ZK-)SNARKs

**SNARKs** are “succinct proofs” that enable to check **correctness of a computation**, with/without **ZK**.

SNARKs from IOPs using hash functions [BCS16]:
Motivation: (ZK-)SNARKs

**SNARKs** are “succinct proofs” that enable to check **correctness of a computation**, with/without **ZK**.

SNARKs from IOPs using hash functions [BCS16]:
- Oracles ← vector commitments (Merkle trees)
SNARKs are “succinct proofs” that enable to check correctness of a computation, with/without ZK.

SNARKs from IOPs using hash functions [BCS16]:
- Oracles ← vector commitments (Merkle trees)
- Non-interactive via $\approx$ Fiat-Shamir transform
Motivation: (ZK-)SNARKs

**Motivation:**

**SNARKs** are “succinct proofs” that enable to check correctness of a computation, with/without ZK.

**SNARKs from IOPs using hash functions [BCS16]:**

- Oracles ← vector commitments (Merkle trees)
- Non-interactive via $\approx$ Fiat-Shamir transform
- Features: **transparent setup, PQ security**

---

I claim that $y = F(x)$

I send you a short proof.
Motivation: (ZK-)SNARKs

SNARKs are “succinct proofs” that enable to check correctness of a computation, with/without ZK.

SNARKs from IOPs using hash functions [BCS16]:

- Oracles ← vector commitments (Merkle trees)
- Non-interactive via ≈ Fiat-Shamir transform
- Features: transparent setup, PQ security
- Key component = IOP of Proximity for codes, with
  - good query/soundness trade-off
  - efficient prover
Motivation: (ZK-)SNARKs

**SNARKs** are “succinct proofs” that enable to check **correctness of a computation**, with/without **ZK**.

SNARKs from IOPs using hash functions [BCS16]:

- Oracles ← vector commitments (Merkle trees)
- Non-interactive via $\approx$ Fiat-Shamir transform
- Features: transparent setup, **PQ security**
- **Key component = IOP of Proximity** for codes, with
  - good query/soundness trade-off
  - efficient prover
- **Size of SNARK:**
  - linear in query complexity of IOP
  - polylog in proof length of IOP
IOPPs for testing proximity to multivariate polynomial codes.

Length is $N$, number of variables is $m$.

Regarding SNARKs applications, constant rate codes $\rightarrow$ shorter proofs ($m = \text{constant}$)

<table>
<thead>
<tr>
<th></th>
<th>Type</th>
<th>Prover</th>
<th>Verifier</th>
<th>Query</th>
<th>Length</th>
<th>Rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>[BBHR18]</td>
<td>RS ($m = 1$)</td>
<td>$&lt; 11N$</td>
<td>$&lt; 11 \log N$</td>
<td>$&lt; 2 \log N$</td>
<td>$&lt; N$</td>
<td>$&lt; \log N$</td>
</tr>
<tr>
<td>This work</td>
<td>RS$^\otimes m$</td>
<td>$&lt; 11N$</td>
<td>$&lt; 11 \log N$</td>
<td>$&lt; 2 \log N$</td>
<td>$&lt; N$</td>
<td>$&lt; \log N$</td>
</tr>
<tr>
<td>This work</td>
<td>RM</td>
<td>$&lt; (2m + 7)N$</td>
<td>$&lt; 2^m \left(\frac{5}{4} + \frac{7}{m}\right) \log N$</td>
<td>$&lt; \frac{2^m \log N}{m}$</td>
<td>$&lt; \frac{N}{2^m - 1}$</td>
<td>$&lt; \frac{\log N}{m}$</td>
</tr>
</tbody>
</table>

(Complexities counted in $\mathbb{F}$-ops and field elements)
Reed-Solomon IOP of Proximity
Reed-Solomon IOPP

**DEF RS code**
Given $L \subset \mathbb{F}$, $d < |L|$, 
$\text{RS}[L,d] = \{f_{|L}: L \to \mathbb{F} \mid f \in \mathbb{F}[X], \deg f < d\}$

The **FRI** protocol is a “Fast Reed-Solomon IOPP” [BBHR18]

Setting:
- $L$ is coset of mult. or add. subgroup of $\mathbb{F}$,
  - $|L| = \text{power of 2}$
- $d = \text{power of 2}$
Reed-Solomon IOPP

**Def RS code**

Given $L \subset F, d < |L|$, 
$RS[L, d] = \{ f|_L : L \to F \mid f \in F[X], \deg f < d \}$

The **FRI** protocol is a “**Fast Reed-Solomon IOPP**” [BBHR18]

Setting:
- $L$ is coset of mult. or add. subgroup of $F$, $|L| = \text{power of 2}$
- $d = \text{power of 2}$

Recursively halve the size of the problem via “random folding”.

---

D/e.scf/f.sc

RS code

The FRI protocol is a “Fast Reed-Solomon IOPP” [BBHR18]

Setting:
- $L$ is coset of mult. or add. subgroup of $F$, $|L| = \text{power of 2}$
- $d = \text{power of 2}$

Recursively halve the size of the problem via “random folding”.

---

D/e.scf/f.sc

RS code
Halve the size of the problem

Assume $\mathbb{F}$ has a multiplicative subgroup $L$ of order $2^n$. The square map $q : x \mapsto x^2$ is 2-to-1 from $L$ to $q(L)$.
Halve the size of the problem

Assume $F$ has a multiplicative subgroup $L$ of order $2^n$. The square map $q : x \mapsto x^2$ is 2-to-1 from $L$ to $q(L)$.

**Goal:** proximity to $\text{RS}[L,d] \rightarrow$ proximity to $\text{RS}[q(L),d/2]$. 
Halve the size of the problem

Assume \( \mathbb{F} \) has a multiplicative subgroup \( L \) of order \( 2^n \).
The square map \( q : x \mapsto x^2 \) is 2-to-1 from \( L \) to \( q(L) \).

**Goal:** proximity to \( \text{RS}[L,d] \rightarrow \text{proximity to } \text{RS}[q(L),d/2] \).

Given **arbitrary** function \( f : L \rightarrow \mathbb{F} \),
Assume $\mathbb{F}$ has a multiplicative subgroup $L$ of order $2^n$. The square map $q : x \mapsto x^2$ is 2-to-1 from $L$ to $q(L)$.

**Goal:** proximity to $\text{RS}[L,d] \rightarrow$ proximity to $\text{RS}[q(L),d/2]$.

Given arbitrary function $f : L \rightarrow \mathbb{F}$,

- Decompose $f$ into two parts:

  $$f(x) = g_0(x^2) + xg_1(x^2), \text{ with } \deg g_i \leq \frac{\deg f}{2}.$$
Assume \( \mathbb{F} \) has a multiplicative subgroup \( L \) of order \( 2^n \). The square map \( q : x \mapsto x^2 \) is 2-to-1 from \( L \) to \( q(L) \).

**Goal:** proximity to \( \text{RS}[L,d] \rightarrow \text{proximity to RS}[q(L),d/2] \).

---

**Given arbitrary** function \( f : L \rightarrow \mathbb{F} \),
- Decompose \( f \) into two parts:
  \[
  f(x) = g_0(x^2) + xg_1(x^2), \text{ with } \deg g_i \leq \frac{\deg f}{2}.
  \]
- For \( z \in \mathbb{F} \), define \( \text{Fold} [f,z] : q(L) \rightarrow \mathbb{F} \) by
  \[
  \text{Fold} [f,z](y) = g_0(y) + zg_1(y).
  \]
Assume $\mathbb{F}$ has a multiplicative subgroup $L$ of order $2^n$. The square map $q : x \mapsto x^2$ is 2-to-1 from $L$ to $q(L)$.

**Goal:** proximity to $\text{RS}[L,d] \rightarrow$ proximity to $\text{RS}[q(L),d/2]$.

Given arbitrary function $f : L \rightarrow \mathbb{F}$,
- Decompose $f$ into two parts:
  
  $$f(x) = g_0(x^2) + xg_1(x^2)$$

  with $\deg g_i \leq \frac{\deg f}{2}$.

- For $z \in \mathbb{F}$, define $\text{Fold}[f,z] : q(L) \rightarrow \mathbb{F}$ by
  $$\text{Fold}[f,z](y) = g_0(y) + zg_1(y).$$

**How to compute $\text{Fold}[f,z]$?**

Any $y \in q(L)$ has 2 distinct square roots $x, -x \in L$. 

Assume $\mathbb{F}$ has a multiplicative subgroup $L$ of order $2^n$. The square map $q : x \mapsto x^2$ is 2-to-1 from $L$ to $q(L)$.

**Goal:** proximity to $\text{RS}[L, d] \rightarrow$ proximity to $\text{RS}[q(L), d/2]$.

Given **arbitrary** function $f : L \to \mathbb{F}$,

- Decompose $f$ into two parts:
  $$f(x) = g_0(x^2) + x g_1(x^2), \text{ with } \deg g_i \leq \frac{\deg f}{2}.$$

- For $z \in \mathbb{F}$, define $\text{Fold}[f, z] : q(L) \to \mathbb{F}$ by
  $$\text{Fold}[f, z](y) = g_0(y) + z g_1(y).$$

**How to compute $\text{Fold}[f, z]$?**

Any $y \in q(L)$ has 2 distinct square roots $x, -x \in L$.

Linear system $\implies g_0(y) = \frac{f(x) + f(-x)}{2}$ and $g_1(y) = \frac{f(x) - f(-x)}{2x}$. 

---

**Halve the size of the problem**
Key properties of folding operators

1. **Completeness:**
   \[ f \in \text{RS}[L,d] \implies \text{Fold} [f,z] \in \text{RS}[q(L),d/2] \text{ for all } z. \]

2. **Local computability:**
   Each entry of \( \text{Fold} [f,z] \) depends on \( l = 2 \) entries of \( f \).
The 3 key properties

Key properties of folding operators

1. Completeness:
   \[ f \in RS[L, d] \implies \text{Fold} [f, z] \in RS[q(L), d/2] \text{ for all } z. \]

2. Local computability:
   Each entry of \( \text{Fold} [f, z] \) depends on \( l = 2 \) entries of \( f \).

3. Distance preservation:
   \[ f \text{ is } \delta \text{-far from } RS[L, d] \implies \text{Fold} [f, z] \text{ is } \delta' \text{-far from } RS[q(L), d/2] \text{ w.h.p over } z \quad (\delta' \approx \delta) \]
Honest prover computes:

\[
\begin{align*}
    f_1 &= \text{Fold}[f_0, z_0] \\
    f_2 &= \text{Fold}[f_1, z_1] \\
    & \quad \vdots \\
    f_r &= \text{Fold}[f_{r-1}, z_{r-1}] \equiv c \in \mathbb{F}
\end{align*}
\nThe prover computes the following steps:

- \( f_0 = f : L \rightarrow \mathbb{F} \)
- \( z_0 \leftarrow \mathbb{F} \)
- \( f_1 = \text{Fold}[f_0, z_0] \)
- \( z_1 \)
- \( f_2 = \text{Fold}[f_1, z_1] \)
- \( z_2 \)
- \( \vdots \)
- \( f_r = \text{Fold}[f_{r-1}, z_{r-1}] \equiv c \in \mathbb{F} \)
Global consistency test:

Sample \( s \in L \) and check

\[
\begin{align*}
  f_1(s^2) &\overset{?}{=} \text{Fold}[f_0, z_0](s^2) \\
  f_2(s^4) &\overset{?}{=} \text{Fold}[f_1, z_1](s^4) \\
  &\vdots \\
  f_r(s^{2^r}) &\overset{?}{=} \text{Fold}[f_{r-1}, z_{r-1}](s^{2^r})
\end{align*}
\]

Final test: \( f_r \equiv c \in \mathbb{F} \)
Honest prover computes:

\[ f_1 = \text{Fold}[f_0, z_0] \]
\[ f_2 = \text{Fold}[f_1, z_1] \]
\[ \vdots \]
\[ f_r = \text{Fold}[f_{r-1}, z_{r-1}] \equiv c \in \mathbb{F} \]

Global consistency test:

Sample \( s \in L \) and check

\[ f_1(s^2) \overset{?}{=} \text{Fold}[f_0, z_0](s^2) \]
\[ f_2(s^4) \overset{?}{=} \text{Fold}[f_1, z_1](s^4) \]
\[ \vdots \]
\[ f_r(s^{2^r}) \overset{?}{=} \text{Fold}[f_{r-1}, z_{r-1}](s^{2^r}) \]

Final test: \( f_r \overset{?}{=} c \in \mathbb{F} \)

Completeness: If \( f \in C \), Verifier accepts.

Soundness: \( \exists \delta_0 \in (0, 1), \forall \delta < \delta_0, \Pr[\text{accept} | f \text{ is } \delta\text{-far}] \approx (1 - \delta) \) (assuming \( |\mathbb{F}| \) is large enough.)
Honest prover computes:

\[ f_1 = \text{Fold}[f_0, z_0] \]

\[ f_2 = \text{Fold}[f_1, z_1] \]

\[ \vdots \]

\[ f_r = \text{Fold}[f_{r-1}, z_{r-1}] \equiv c \in \mathbb{F} \]

Global consistency test:

Sample \( s \in L \) and check

\[ f_1(s^2) \overset{?}{=} \text{Fold}[f_0, z_0](s^2) \]

\[ f_2(s^4) \overset{?}{=} \text{Fold}[f_1, z_1](s^4) \]

\[ \vdots \]

\[ f_r(s^{2^r}) \overset{?}{=} \text{Fold}[f_{r-1}, z_{r-1}](s^{2^r}) \]

Final test: \( f_r \overset{?}{=} c \in \mathbb{F} \)

Completeness: If \( f \in C \), Verifier accepts.

Soundness: \( \exists \delta_0 \in (0, 1), \forall \delta < \delta_0, \text{Pr}[\text{accept } | f \text{ is } \delta\text{-far}] \approx (1 - \delta)^t. \) (assuming \( |\mathbb{F}| \) is large enough.)
How to fold multivariate polynomials
The tensor structure enables to fold along **one variable at a time**.
The tensor structure enables to fold along **one variable at a time**.

- $\text{RS}^\otimes m \rightarrow \text{RS}^\otimes m - 1$: 

...
Folding tensor product of RS codes

The tensor structure enables to fold along one variable at a time.

- RS$^\otimes m \rightarrow$ RS$^\otimes m-1$:
  - Write $f : \prod_{i=1}^{m} L_i \rightarrow \mathbb{F}$ as
    $$f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m)$$
Folding tensor product of RS codes

The tensor structure enables to fold along one variable at a time.

- RS$^\otimes m \rightarrow$ RS$^\otimes m - 1$:
  \[ (q : x \mapsto x^2) \]
  
  - Write $f : \prod_{i=1}^{m} L_i \rightarrow \mathbb{F}$ as
    \[ f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m) \]
  
  - For $z \in \mathbb{F}$, define $\text{Fold} \ [f, z] : q(L_1) \times \prod_{i=2}^{m} L_i \rightarrow \mathbb{F}$ by
    \[ \text{Fold} \ [f, z] (y, x) = g_0(y, x) + zg_1(y, x) \]
The tensor structure enables to fold along **one variable at a time**.

- **RS**$^\otimes m \rightarrow$ RS$^\otimes m-1$:
  - Write $f : \prod_{i=1}^{m} L_i \rightarrow \mathbb{F}$ as
    $$f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m)$$
  - For $z \in \mathbb{F}$, define $\text{Fold}[f, z] : q(L_1) \times \prod_{i=2}^{m} L_i \rightarrow \mathbb{F}$ by
    $$\text{Fold}[f, z](y, x) = g_0(y, x) + zg_1(y, x)$$
  - After $\log d$ rounds, expected $x_1$-degree = 0
Folding tensor product of RS codes

The tensor structure enables to fold along one variable at a time.

- **RS$^{\otimes m}$ → RS$^{\otimes m-1}$:**
  - Write $f : \prod_{i=1}^{m} L_i \rightarrow \mathbb{F}$ as
    \[
    f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m)
    \]
  - For $z \in \mathbb{F}$, define $\text{Fold}[f, z] : q(L_1) \times \prod_{i=2}^{m} L_i \rightarrow \mathbb{F}$ by
    \[
    \text{Fold}[f, z](y, x) = g_0(y, x) + zg_1(y, x)
    \]
  - After $\log d$ rounds, expected $x_1$-degree $= 0$

- **RS$^{\otimes m-1}$ → RS$^{\otimes m-2}$**, starting with $f : \prod_{i=2}^{m} L_i \rightarrow \mathbb{F}$
Folding tensor product of RS codes

The tensor structure enables to fold along one variable at a time.

- $\text{RS}^\otimes m \rightarrow \text{RS}^\otimes m-1$:
  - Write $f : \prod_{i=1}^{m} L_i \rightarrow \mathbb{F}$ as
    \[
    f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1 g_1(x_1^2, x_2, \ldots, x_m)
    \]
  - For $z \in \mathbb{F}$, define $\text{Fold}[f, z] : q(L_1) \times \prod_{i=2}^{m} L_i \rightarrow \mathbb{F}$ by
    \[
    \text{Fold}[f, z](y, x) = g_0(y, x) + z g_1(y, x)
    \]
  - After $\log d$ rounds, expected $x_1$-degree $= 0$

- $\text{RS}^\otimes m-1 \rightarrow \text{RS}^\otimes m-2$, starting with $f : \prod_{i=2}^{m} L_i \rightarrow \mathbb{F}$

- ... $\rightarrow \text{RS code of dimension 1}$
Folding tensor product of RS codes

The tensor structure enables to fold along **one variable at a time**.

- **RS**$^\otimes m \rightarrow$ RS$^\otimes m-1$:
  
  \( q : x \mapsto x^2 \)
  
  > Write \( f : \prod_{i=1}^{m} L_i \rightarrow \mathbb{F} \) as
  
  \[
  f(x_1, x_2, \ldots, x_m) = g_0(x_1^2, x_2, \ldots, x_m) + x_1g_1(x_1^2, x_2, \ldots, x_m)
  \]
  
  > For \( z \in \mathbb{F} \), define Fold \([f, z] : q(L_1) \times \prod_{i=2}^{m} L_i \rightarrow \mathbb{F} \) by
  
  \[
  \text{Fold} [f, z] (y, x) = g_0(y, x) + zg_1(y, x)
  \]
  
  > After \( \log d \) rounds, expected \( x_1 \)-degree = 0

- **RS**$^\otimes m-1 \rightarrow$ RS$^\otimes m-2$, starting with \( f : \prod_{i=2}^{m} L_i \rightarrow \mathbb{F} \)

- ... \rightarrow RS code of dimension 1

- **Completeness** ✔️  **Local computability** ✔️  **Distance preservation** ✔️
In the total degree case, we fold along every variable at the same time.
In the total degree case, we fold along *every variable at the same time.*
In the total degree case, we fold along every variable at the same time.

**Multivariate decomposition**

Let \( f(X) \in \mathbb{F}[X_1, \ldots, X_m] \). There is a unique sequence of polynomials \((g_u)_{u \in \{0,1\}^m}\) such that

\[
f(X) = \sum_{u \in \{0,1\}^m} X^u g_u(X_1^2, \ldots, X_m^2), \quad \deg g_u \leq \left\lfloor \frac{\deg f - w_H(u)}{2} \right\rfloor
\]
In the total degree case, we fold along **every variable at the same time**.

**Multivariate decomposition**

Let $f(\mathbf{X}) \in \mathbb{F}[X_1, \ldots, X_m]$.
There is a unique sequence of polynomials $(g_u)_{u \in \{0,1\}^m}$ such that

$$f(\mathbf{X}) = \sum_{u \in \{0,1\}^m} \mathbf{X}^u g_u(X_1^2, \ldots, X_m^2), \quad \deg g_u \leq \left\lfloor \frac{\deg f - \omega_H(u)}{2} \right\rfloor$$

The folding of $f : L^m \to \mathbb{F}$ is a function $\text{Fold} [f, z] : q(L)^m \to \mathbb{F}$ defined using the $g_u$'s.

Reduce $\text{RM}[L, d, m] \rightarrow \text{RM}[q(L), d/2, m]$.

**Subtlety:** need to be careful about the distinct degree bounds on the $g_u$'s.
In the total degree case, we fold along every variable at the same time.

**Multivariate decomposition**

Let \( f(\mathbf{X}) \in \mathbb{F}[X_1, \ldots, X_m] \).

There is a unique sequence of polynomials \((g_u)_{u \in \{0,1\}^m}\) such that

\[
    f(\mathbf{X}) = \sum_{u \in \{0,1\}^m} X^u g_u(X_1^2, \ldots, X_m^2), \quad \deg g_u \leq \left\lfloor \frac{\deg f - w_H(u)}{2} \right\rfloor
\]

The folding of \( f : L^m \rightarrow \mathbb{F} \) is a function \( \text{Fold}[f, z] : q(L)^m \rightarrow \mathbb{F} \) defined using the \( g_u \)'s.

Reduce \( \text{RM}[L, d, m] \rightarrow \text{RM}[q(L), d/2, m] \).

Subtlety: need to be careful about the distinct degree bounds on the \( g_u \)'s.

- ✔ Completeness
- ✔ Local computability (with \( l = 2^m \))
- ✔ Distance preservation
What we have:

- Inspired by FRI protocol for RS-IOPP, we give concrete IOPPs for $\text{RS}^{\otimes m}$ and RM codes with similar parameters.

**Theorem [Augot-Bordage-Nardi’21]**

$\text{RS}[L, d]^{\otimes m}$ has an IOPP $(P, V)$ satisfying

\[
\begin{align*}
\text{# rounds} & = m \log d \\
\text{# queries} & = 2m \log d + 1 \\
\text{prover time} & \leq 11|L^m| \\
\text{verifier time} & \leq 11m \log d \\
\text{proof length} & < |L^m|
\end{align*}
\]

**Theorem [Augot-Bordage-Nardi’21]**

$\text{RM}[L, d, m]$ has an IOPP $(P, V)$ satisfying

\[
\begin{align*}
\text{# rounds} & = \log d \\
\text{# queries} & = 2^m \log d + 1 \\
\text{prover time} & < (2m + 7)|L^m| \\
\text{verifier time} & < 2^m(\frac{5}{4}m + 7)(\log |L|) \\
\text{proof length} & < |L^m|/(2^m - 1)
\end{align*}
\]

Future directions:

- Narrow the gap between theoretically feasible and practical IOPP with sublogarithmic query complexity? (in theory, $O(1)$ queries)
- Practical IOPP for $C^{\otimes m}$ where $C$ is a generic linear code?

Thank you!
What we have:

- Inspired by FRI protocol for RS-IOPP,
  we give concrete IOPPs for $\text{RS}^\otimes_m$ and RM codes with similar parameters.
- Folding-based IOPP framework definition of sequence of codes with folding operators
Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP, we give concrete IOPPs for $\text{RS}^{\otimes m}$ and RM codes with similar parameters.
- Folding-based IOPP framework defines sequence of codes with folding operators.
  - also yields to IOPP for AG codes (next talk)
Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP, we give concrete IOPPs for $\text{RS}^{\otimes m}$ and RM codes with similar parameters.
- Folding-based IOPP framework definition of sequence of codes with folding operators
  - also yields to IOPP for AG codes (next talk)
Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP, we give concrete IOPPs for $\text{RS}^{\otimes m}$ and RM codes with similar parameters.
- Folding-based IOPP framework definition of sequence of codes with folding operators also yields to IOPP for AG codes (next talk)

Future directions: narrow the gap between theoretically feasible and practical

- Practical IOPP with sublogarithmic query complexity? (in theory, $O(1)$ queries)
- Practical IOPP for $\text{C}^{\otimes m}$ where $\text{C}$ is a generic linear code?
Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP, we give concrete IOPPs for $\text{RS}^{\otimes m}$ and RM codes with similar parameters.
- Folding-based IOPP framework also yields to IOPP for AG codes (next talk)

Future directions: narrow the gap between theoretically feasible and practical

- Practical IOPP with sublogarithmic query complexity? (in theory, $O(1)$ queries)
- Practical IOPP for $C^{\otimes m}$ where $C$ is a generic linear code?
Conclusion

What we have:

- Inspired by FRI protocol for RS-IOPP, we give concrete IOPPs for $\text{RS}^\otimes m$ and RM codes with similar parameters.
- Folding-based IOPP framework definition of sequence of codes with folding operators

Future directions: narrow the gap between theoretically feasible and practical

- Practical IOPP with sublogarithmic query complexity? (in theory, $O(1)$ queries)
- Practical IOPP for $C^\otimes m$ where $C$ is a generic linear code?

Thank you!