FAST REED-SOLOMON INTERACTIVE ORACLE PROOFS of PROXIMITY

Tiago Martins

Formerly an Aerospace student @IST
Currently, a Researcher @Three Sigma

Venture builder firm focused on blockchain engineering, research, and investment.

Mission

Advance the adoption of blockchain technology. Contribute for the healthy development of crypto/web3.

PROBLEM SETTING

A non-necessarily honest prover wants to indirectly settle some data in the blockchain.

A verifier must check whether that data corresponds to honest computations and transactions.

PROBLEM SETTING

Both the prover and the verifier know that valid data corresponds to the evaluation table of a low-degree polynomial.

This happens because valid computations are very structured and follow strict low-degree constraints.

PROBLEM EXAMPLE

Trace index	Group element x	Trace polynomial $f(x)$
0	$x_0 = g^0$	3
1	$x_1 = g^1$	9
2	$x_2 = g^2$	81
3	$x_3 = g^3$	6561
4	$x_4 = g^4$	43046721
5	$x_5 = g^5$	1853020188851841

$$f(\underbrace{x_{k+1}}_{g\,x_k}) = (f(x_k))^2 \quad \text{for} \quad x_k \in \{x_0, x_1, x_2, x_3, x_4\}$$

PROBLEM EXAMPLE

$$f(\underbrace{x_{k+1}}_{g\,x_k}) = (f(x_k))^2 \quad \text{for} \quad x_k \in \{x_0, x_1, x_2, x_3, x_4\}$$

which implies that

$$f(g\,x) - (f(x))^2 = 0 \quad \text{for} \quad x \in \{x_0, x_1, x_2, x_3, x_4\}$$

which in turn implies that

$$p_0(x) = \frac{f(g\,x) - (f(x))^2}{(x - x_0)(x - x_1)(x - x_2)(x - x_3)(x - x_4)}$$

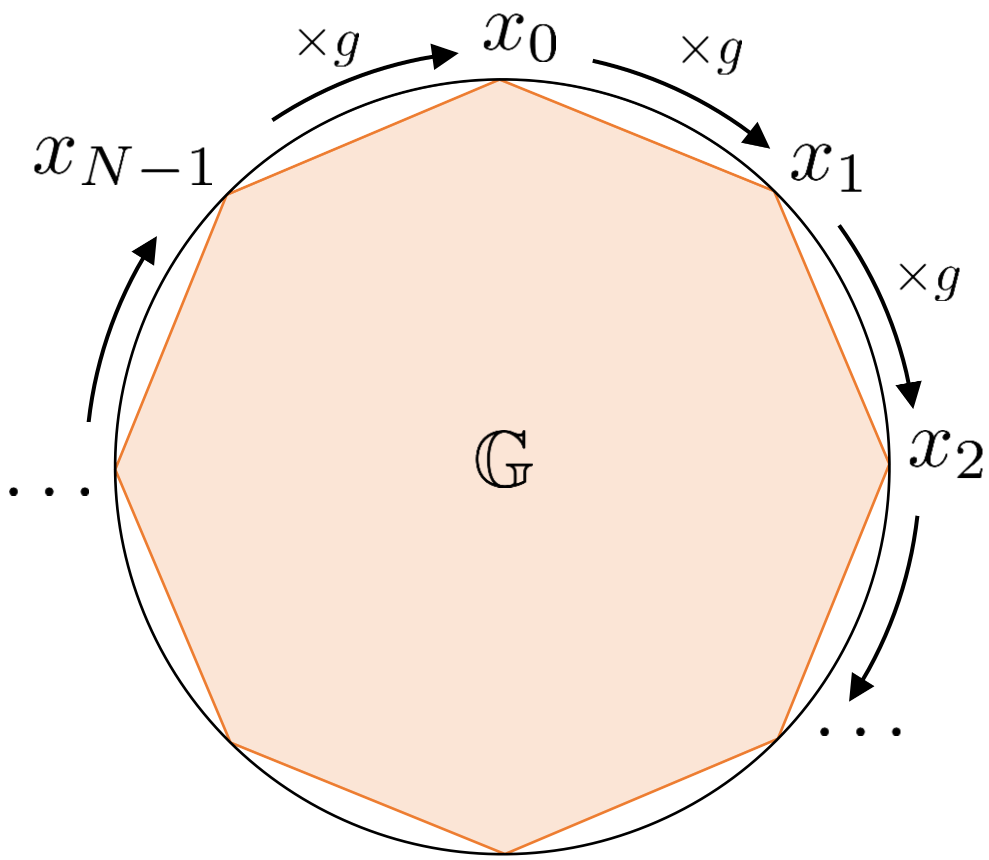
is a polynomial of degree $\deg(p_0) = 2\deg(f) - 5$.

POLYNOMIAL

$$p(x) = \sum_{n=0}^{N-1} c_n x^n$$

$$\underbrace{\begin{bmatrix} p(x_0) \\ p(x_1) \\ \vdots \\ p(x_{N-1}) \end{bmatrix}}_{\mathbf{p}} = \begin{bmatrix} 1 & x_0 & \dots & x_0^{N-1} \\ 1 & x_1 & \dots & x_1^{N-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_{N-1} & \dots & x_{N-1}^{N-1} \end{bmatrix} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{bmatrix}}_{\mathbf{c}}$$

FAST FOURIER TRANSFORM TO EVALUATE POLYNOMIALS

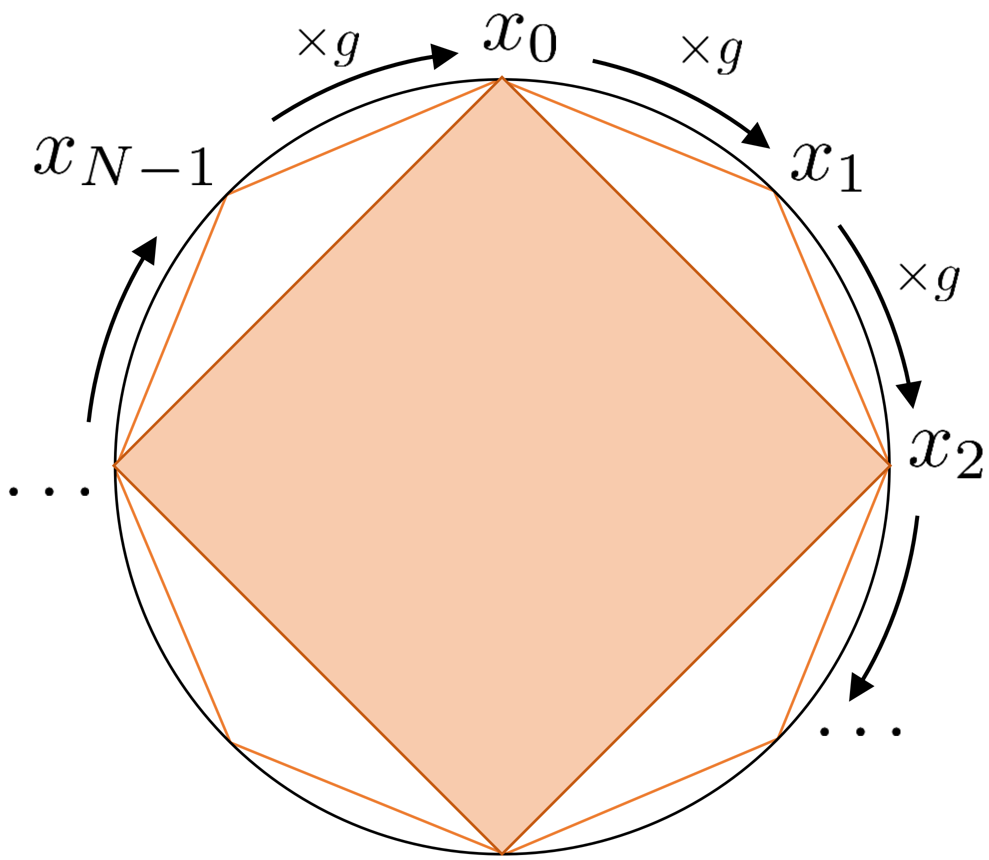


$x \in \mathbb{G}$

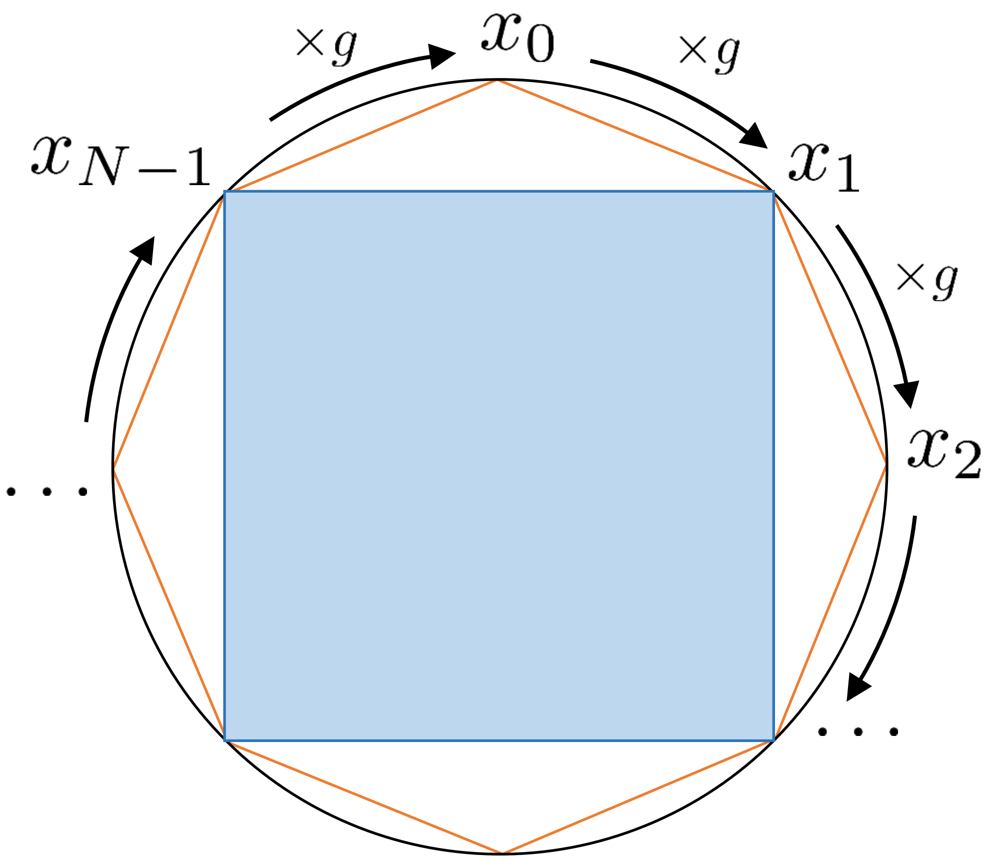
$$\mathbf{p} = \mathrm{FFT}(\mathbf{c})$$

$$\mathbf{c} = \mathrm{iFFT}(\mathbf{p})$$

FAST FOURIER TRANSFORM to EVALUATE POLYNOMIALS



$x \in \mathbb{G}$



HANDS-ON!

Files:
Base.ipynb
FFT for function evaluation.ipynb

https://github.com/tiagomartins-threesigma/SINFO2023FRI4STARKs

DEGREE RESPECTING PROJECTION

$$
\begin{aligned}
\mathbf{c} &= [c_0, && c_1, && c_2, && c_3, && \ldots, && c_{N-2}, && c_{N-1}] \\
\mathbf{c}_{\text{even}} &= [c_0, && && c_2, && && \ldots, && c_{N-2} &&] \\
\mathbf{c}_{\text{odd}} &= [&& c_1, && && c_3, && \ldots, && && c_{N-1}]
\end{aligned}
$$

$$
\begin{aligned}
\text{len}(\mathbf{c}) &= N \\
\text{len}(\mathbf{c}_{\text{even}}) &= N/2 \\
\text{len}(\mathbf{c}_{\text{odd}}) &= N/2
\end{aligned}
$$

DEGREE RESPECTING PROJECTION

$$
\begin{array}{rcllllllll}
\mathbf{c} & = & [c_0, & c_1, & c_2, & c_3, & \ldots, & c_{N-2}, & c_{N-1} &] \\
\mathbf{c}_{\text{even}} & = & [c_0, & & c_2, & & \ldots, & c_{N-2} & &] \\
\mathbf{c}_{\text{odd}} & = & [& c_1, & & c_3, & \ldots, & & c_{N-1} &] \\
\mathbf{c}_{\text{new}} & = & [c_0 z_{\text{even}} + & c_1 z_{\text{odd}}, & c_2 z_{\text{even}} + & c_3 z_{\text{odd}}, & \ldots, & c_{N-2} z_{\text{even}} + & c_{N-1} z_{\text{odd}} &]
\end{array}
$$

$$\text{len}(\mathbf{c}_{\text{new}}) = N/2$$

$$\mathbf{c}_{\text{new}} = \mathbf{c}_{\text{even}}\, z_{\text{even}} + \mathbf{c}_{\text{odd}}\, z_{\text{odd}}$$

```
c_new = c[::2]*z_even + c[1::2]*z_odd
```

Desmos

FRI COMMIT PHASE

For each FRI layer $i \geq 0$ such that $d_i > 1$:

1. the verifier selects random $z_{i,\text{even}}$ and $z_{i,\text{odd}}$,
2. the prover defines $p_{i+1}(x)$ through the degree respecting projection $\mathbf{c}_{i+1} = \mathbf{c}_{i,\text{even}}\, z_{i,\text{even}} + \mathbf{c}_{i,\text{odd}}\, z_{i,\text{odd}}$,
3. the prover commits to the polynomial evaluation $\mathbf{p}_{i+1} = \texttt{FFT}(\mathbf{c}_{i+1})$,
4. the degree bound reduces as $d_{i+1} = d_i/2$.

At the end, the prover commits to a constant C allegedly euqual to the last polynomial.

HANDS-ON!

Files:
FRI Commit.ipynb

https://github.com/tiagomartins-threesigma/SINFO2023FRI4STARKs

DRP wo/ COEFFICIENTS

$$p_i(x) = \sum_{n=0}^{N-1} c_{i,n}\, x^n \qquad p_i(-x) = \sum_{n=0}^{N-1} (-1)^n\, c_{i,n}\, x^n$$

$$p_{i+1}(x) = \sum_{n=0}^{N/2-1} c_{i+1,n}\, x^n \quad \text{where} \quad \mathbf{c}_{i+1} = \mathbf{c}_{i,\text{even}}\, z_{i,\text{even}} + \mathbf{c}_{i,\text{odd}}\, z_{i,\text{odd}}$$

$$\boxed{p_{i+1}(x^2) = \underbrace{\frac{p_i(x) + p_i(-x)}{2}}_{g(x^2)}\, z_{i,\text{even}} + \underbrace{\frac{p_i(x) - p_i(-x)}{2x}}_{h(x^2)}\, z_{i,\text{odd}}}$$

FRI QUERY PHASE

The verifier samples a random x. The verifier queries for $p_0(x)$, and, subsequently, for each FRI layer $i \geq 0$ such that $d_i > 1$:

1. the verifier queries for $p_i(-x)$ and $p_{i+1}(x^2)$,
2. the verifier computes $g(x^2)$ and $h(x^2)$ from $p_i(x)$ and $p_i(-x)$,
3. the verifier checks whether $p_{i+1}(x^2) = g(x^2)\, z_{i+1,\text{even}} + h(x^2)\, z_{i+1,\text{odd}}$,
4. set $x \leftarrow x^2$.

At the end, the verifier checks whether the last value equals the constant C.

HANDS-ON!

Files:
FRI Query.ipynb

https://github.com/tiagomartins-threesigma/SINFO2023FRI4STARKs

Three Sigma

info@threesigma.xyz

threesigma.xyz