FRI-based SNARKs

Verifiable computing via proximity tests to Reed-Solomon codes

Hugo Delavenne

Séminaire IAA / IMATH

Jeudi 12 mars 2026

1. SNARKs
2. AIR arithmetization
3. Fast Reed-Solomon IOPP

1 SNARKs

2 AIR arithmetization

3 Fast Reed-Solomon IOPP

- Why verifiable computing?
  - Verifiable computing
  - Applications
- Cryptographic tools
  - Interactive Proofs
  - Fiat-Shamir
  - Oracle access
  - FRI-based SNARK workflow

Computing: Given A and x, compute $y = A(x)$

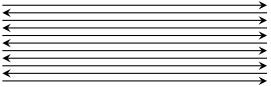
Verifiable computing: Given A and x, compute $y = A(x)$ and π: "$y = A(x)$"

- π is short
- π is fast to verify

- Blockchains: short proof on chain
- Cryptography: e.g. untrusted dealer, or PREON signature
- Delegated computing
- Verifiable AI
- Correct video game execution (correct emulation for speedrun)

SNARK means **S**uccinct **N**on-interactive **AR**gument of **K**nowledge.

But we will see Interactive Proofs:

$\mathcal{P}$rover  $\mathcal{V}$erifier

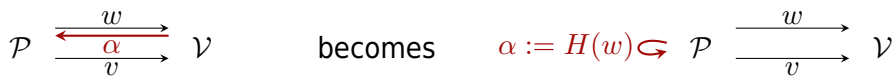
- made non-interactive with **Fiat-Shamir**
- made succinct with **oracle accesses**

Definition **Cryptographic hash function**

A cryptographic **hash function** $H : \{0,1\}^* \to \{0,1\}^{|H|}$

- looks **injective**: $\mathcal{P}$ cannot find collisions
- looks **random**: $\mathcal{P}$ cannot find input to get desired output

Fiat-Shamir replaces $\mathcal{V}$'s randomness by hash of previous messages:

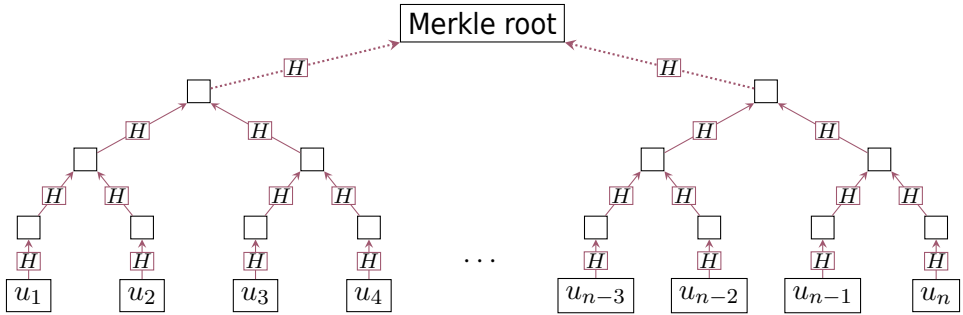


[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive Oracle Proofs.
In *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part II 14*, pages 31–60. Springer, 2016

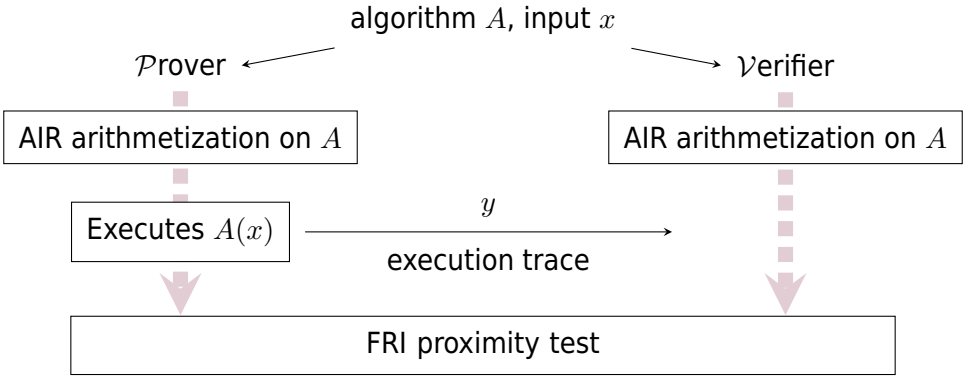
Definition **Oracle access**

$\mathcal{P}$ provides $\mathcal{V}$ **oracle access** to $u \in \mathbb{F}^n$ by giving **black-box** access to coordinates

In practice, $\mathcal{P}$ provides the root of a **Merkle tree**.



Merkle root created in $O(n)$, Merkle branch opened in $O(\log n)$



Completeness: if $\boldsymbol{A(x) = y}$ then $\mathbb{P}(\mathcal{V}^{\leftrightarrow\mathcal{P}}(A, x, y) \text{ accepts}) = 1$

Soundness: if $\boldsymbol{A(x) \neq y}$ then for any $\mathcal{P}$, $\mathbb{P}(\mathcal{V}^{\leftrightarrow\mathcal{P}}(A, x, y) \text{ accepts}) \leqslant s$

1 SNARKs

2 AIR arithmetization

3 Fast Reed-Solomon IOPP

- Square Fibonacci
  - A simple algorithm
  - Constraint polynomials
- Generic arithmetization
  - More advanced constraints
  - AIR arithmetization
- Reduction to proximity testing
  - Reed-Solomon codes
  - Why testing proximity suffices?

Consider the algorithm

```
u = 1;
v = 1;
for i = 2 to T {
    tmp = u;
    u = u^2 + v^2;
    v = tmp;
}
return u;
```

It computes $u_{\mathtt{T}}$ defined by

$$\begin{cases} u_1 = v_1 = 1 \\ u_{i+1} = u_i^2 + v_i^2 \\ v_{i+1} = u_i. \end{cases}$$

Let $G = \langle g \rangle$ be of order $\mathtt{T}$ and $u(X), v(X)$ such that

$$u(g^i) = u_i,$$
$$v(g^i) = v_i.$$

$u(X)$, $v(X)$ are **execution trace polynomials**

Remark that $u(g \cdot g^i) = u_{i+1}$ and same for v.

Let

- $Q_u(X_u, X_v, Y_u, Y_v) := Y_u - X_u^2 - X_v^2$
- $Q_v(X_u, X_v, Y_u, Y_v) := Y_v - X_u$

Define $Q \circ (u, v)(X) := Q(u(X), v(X), u(gX), v(gX))$ so that

$$Q_u \circ (u, v)(X) = u(gX) - u(X)^2 - v(X)^2 \dashrightarrow u_{i+1} - u_i^2 - v_i^2$$
$$Q_v \circ (u, v)(X) = v(gX) - u(X) \dashrightarrow v_{i+1} - u_i$$

If the execution is valid, $Q_u \circ (u, v)(X)$ and $Q_v \circ (u, v)(X)$ cancel on G, i.e.

$$\exists f_u(X), f_v(X), \quad \begin{cases} Q_u \circ (u, v)(X) = f_u(X) \prod (X - g^t) \\ Q_v \circ (u, v)(X) = f_v(X) \prod (X - g^t) \end{cases}$$

Remark that $\prod (X - g^t) = X^{\mathrm{T}} - 1$ can be computed efficiently.

- For instructions only applied at steps $t \in S$, there are **selector polynomials**:

$$Q \circ (R_1, ..., R_m)(X) \cdot \prod_{t \notin S} (X - g^t)$$

- When there are `if` and `while`, we can use an **instruction pointer**
- There are libraries to turn any Rust program into RISC5 assembly, and arithmetize it: **RISC0**

If A is on m registers $R_1, ..., R_m$, AIR(A) outputs **constraint polynomials**

$$Q_1, ..., Q_\ell \in \mathbb{F}[X_1, ..., X_m, Y_1, ..., Y_m]$$

such that

$$\forall t, \begin{array}{|c|}\hline R_1(t) \\ \hline \vdots \\ \hline R_m(t) \\ \hline \end{array} \xmapsto{I_t} \begin{array}{|c|}\hline R_1(t+1) \\ \hline \vdots \\ \hline R_m(t+1) \\ \hline \end{array} \iff \forall t, \begin{cases} Q_1 \circ (R_1, ..., R_m)(g^t) = 0 \\ \vdots \\ Q_\ell \circ (R_1, ..., R_m)(g^t) = 0 \end{cases}$$

$$\iff \exists f_1, ..., f_\ell, \begin{cases} Q_1 \circ (R_1, ..., R_m)(X) = f_1(X)(X^T - 1) \\ \vdots \\ Q_\ell \circ (R_1, ..., R_m)(X) = f_\ell(X)(X^T - 1) \end{cases}$$

Definition **Relative Hamming distance**

Let $u, v \in \mathbb{F}^n$.

$$\Delta(u, v) := \frac{1}{n}|\{i \in [n] \mid u_i \neq v_i\}|$$

A linear **error correcting code** is a linear subspace of $\mathbb{F}^n$.

Definition **Reed-Solomon code**

Let $\mathcal{L} \subseteq \mathbb{F}$ and $k < |\mathcal{L}|$.

$$\mathsf{RS}[\mathcal{L}, k] := \{f : \mathcal{L} \to \mathbb{F} \mid f \in \mathbb{F}[X]_{\leqslant k-1}\}$$

Why testing proximity suffices?

Let $\boldsymbol{R_1}, ..., \boldsymbol{R_m}, \boldsymbol{f_1}, ..., \boldsymbol{f_\ell} : \mathcal{L} \to \mathbb{F}$.

Assume that **the computation is not valid**, i.e for any $\tilde{R}_1, ..., \tilde{R}_m, \tilde{f}_1, ..., \tilde{f}_\ell \in \mathbb{F}[X]$,

$$\exists i, Q_i \circ (\tilde{R}_1, ..., \tilde{R}_m)(X) \neq \tilde{f}_i(X)(X^T - 1).$$

Then either

- $\mathbb{P}_{x \in \mathcal{L}}(Q_i \circ (\boldsymbol{R_1}, ..., \boldsymbol{R_m})(x) = \boldsymbol{f_i}(x)(x^T - 1)) \leqslant \frac{T \deg Q_i}{|\mathcal{L}|} + \delta$, → test a few values
- $\Delta(\boldsymbol{R_j}, \mathsf{RS}[\mathcal{L}, T]) \geqslant \delta$ for some j,
- $\Delta(\boldsymbol{f_i}, \mathsf{RS}[\mathcal{L}, T(\deg Q_i - 1)]) \geqslant \delta$.

} proximity tests!

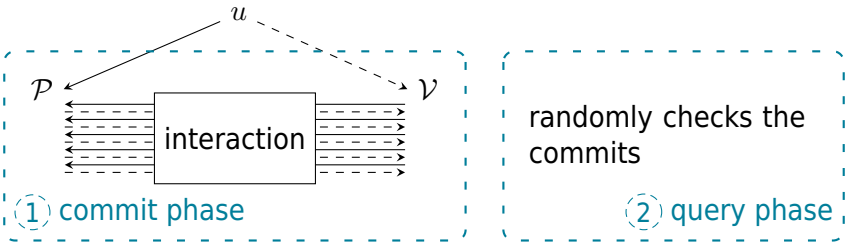
[Bor22] Sarah Bordage. *Efficient protocols for testing proximity to algebraic codes*. **Theses, Institut Polytechnique de Paris, June 2022**

1 SNARKs

2 AIR arithmetization

3 Fast Reed-Solomon IOPP

- Protocol
  - Interactive Oracle Proof of Proximity
  - Folding Reed-Solomon codewords
  - FRI protocol
- Analysis
  - Security analysis
  - Completeness and soundness
  - FRI complexity



Completeness: if $u \in C$ then $\mathbb{P}(\mathcal{V}^{u,\leftrightarrow\mathcal{P}} \text{ accepts}) = 1$

Soundness: if $\Delta(u, C) > \delta$ then for any $\mathcal{P}$, $\mathbb{P}(\mathcal{V}^{u,\leftrightarrow\mathcal{P}} \text{ accepts}) \leqslant s$

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive Oracle Proofs.
In *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part II 14*, pages 31–60. Springer, 2016

Idea: Test simultaneously even and odd parts

$$f(X) =: f_{\mathsf{even}}(X^2) + X f_{\mathsf{odd}}(X^2).$$

Definition **Folding**

Let $f : \mathcal{L}_0 \to \mathbb{F}$ and $\alpha \in \mathbb{F}$. Let $\mathcal{L}_1 := \{x^2 \mid x, -x \in \mathcal{L}_0\}$.

$$\mathsf{Fold}[f, \alpha](X^2) := f_{\mathsf{even}}(X^2) + \alpha f_{\mathsf{odd}}(X^2) = \frac{f(X) + f(-X)}{2} + \alpha \frac{f(X) - f(-X)}{2X}$$

Fold satisfies **locality**: $\mathcal{V}$ computes $\mathsf{Fold}[f, \alpha](x^2)$ with queries to $f(x)$ and $f(-x)$.

Prover will perform successive foldings on $\mathcal{L}_{i+1} := \{x^2 \mid x, -x \in \mathcal{L}_i\}$.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon Interactive Oracle Proofs of Proximity. **In *45th international colloquium on automata, languages, and programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018**

Protocol starts with $f_0 : \mathcal{L}_0 \to \mathbb{F}$.

Commit phase: for $i \in [\log k]$,

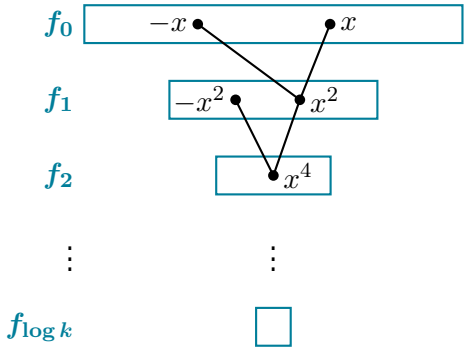
- $\mathcal{V}$ sends random $\alpha_i \in \mathbb{F}$
- $\mathcal{P}$ sends oracle to $f_i : \mathcal{L}_i \to \mathbb{F}$
  if $\mathcal{P}$ is honest, $f_i = \mathsf{Fold}[f_{i-1}, \alpha_i]$

Query phase: repeat m times

- $\mathcal{V}$ chooses random $x \in \mathcal{L}_0$
- for $i \in [\log k]$, $\mathcal{V}$ checks that

$$f_i(x^{2^i}) \stackrel{?}{=} \mathsf{Fold}[f_{i-1}, \alpha_i](x^{2^i})$$

- $\mathcal{V}$ checks that $f_{\log k} \stackrel{?}{\in} \mathsf{RS}[\mathcal{L}_{\log k}, 1]$



There are two possible issues:

A **commit error** is when

$$\Delta(\mathsf{Fold}[f_i, \alpha_{i+1}], C_{i+1}) < \Delta(f_i, C_i),$$

with $C_i := \mathsf{RS}[\mathcal{L}_i, \frac{k}{2^i}]$

Proposition

It happens w.p. over α_i

$$\leqslant \frac{10^7 |\mathcal{L}_0|^{3.5}}{k^{1.5} |\mathbb{F}|}$$

A **query error** is when

$$f_i \neq \mathsf{Fold}[f_{i-1}, \alpha_i]$$

but for the chosen x

$$f_i(x^{2^i}) = \mathsf{Fold}[f_{i-1}, \alpha_i](x^{2^i})$$

Proposition

If no commit error, it happens w.p.

$$\leqslant \left(1 - \min\left(\delta, 1 - \sqrt{k/|\mathcal{L}_0|}\right)\right)^m.$$

[BCI+23] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity Gaps for Reed–Solomon Codes. ***J. ACM*, 70(5), October 2023**

Proposition **FRI completeness**

If $f_0 \in \mathbf{RS}[\mathcal{L}_0, k]$ then $\mathcal{V}$ accepts with probability 1.

Theorem **FRI soundness**

If $\boldsymbol{\Delta}(f_0, \mathbf{RS}[\mathcal{L}_0, k]) > \delta$ then for any $\tilde{\mathcal{P}}$ and $\varepsilon > 0$, $\mathcal{V}$ accepts with probability

$$\leqslant \frac{10^7 |\mathcal{L}_0|^{3.5} \log k}{k^{1.5} |\mathbb{F}|} + \left(1 - \min\left(\delta, 1 - \sqrt{k/|\mathcal{L}_0|}\right)\right)^m$$

For security λ: in practice $\lambda = 128$

- $|\mathbb{F}| > 2^\lambda \cdot \frac{10^7 |\mathcal{L}_0|^{3.5} \log k}{k^{1.5}}$ $\quad |\mathbb{F}| > 2^{194}$
- $m > \frac{\lambda}{\min\left(\delta, 1 - \sqrt{k/|\mathcal{L}_0|}\right)}$ $\quad m > 200$

[BCI+23] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity Gaps for Reed–Solomon Codes. *J. ACM*, 70(5), October 2023

FRI protocol for RS$[\mathcal{L}_0, k]$ at distance $\delta < 1 - \sqrt{k/n}$, with $n = |\mathcal{L}_0|$ has complexities

- Encoding: $O(\boldsymbol{n \log n})$
- Prover complexity: $< 8\boldsymbol{n}$
- Verifier complexity and proof size: $< 2\frac{\mathbf{1}}{\boldsymbol{\delta}}\boldsymbol{\lambda \log k \log n}$

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon Interactive Oracle Proofs of Proximity.
In *45th international colloquium on automata, languages, and programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018

1 SNARKs

2 AIR arithmetization

3 Fast Reed-Solomon IOPP

▷ Conclusion

Advantages and drawbacks

- \+ Very generic
- \+ No trusted setup
- \- Field restrictive
- \- Prover not linear because of encoding

More recent IOPP

- ▷ BaseFold (2023): “Foldable codes”, no field restriction
- ▷ STIR & WHIR (2024): RS codes, better query complexity
- ▷ **Flowering** (2025): some LDPC codes, no field restriction → **for next time!**