Introduction to STARKs

Alan Szepieniec
艾伦·佘丕涅茨
alan@neptune.cash

neptune

https://neptune.cash/

Triton VM

https://triton-vm.org/

https://asz.ink/presentations/2025-09-18-Introduction-to-STARKs.pdf

Table of Contents

Table of Contents

Motivation

Virtual Machine $\longleftrightarrow$ Fixed Circuit

Motivation

Virtual Machine $\longleftrightarrow$ Fixed Circuit

Virtual Machine

- versatile

  high one-time cost

  low per-application cost

  slower + more resources required

Fixed Circuit

- optimizable

  high per-application cost

  faster + fewer resources required

Motivation

Virtual Machine $\longleftrightarrow$ Fixed Circuit

Virtual Machine

- versatile
  high one-time cost
  low per-application cost
  slower + more resources required
- one prover, one verifier, many programs

Fixed Circuit

- optimizable
  high per-application cost
  faster + fewer resources required
- one prover and one verifier *for each* application

Motivation

Virtual Machine $\longleftrightarrow$ Fixed Circuit

Virtual Machine

- versatile
  - high one-time cost
  - low per-application cost
  - slower + more resources required
- one prover, one verifier, many programs
- examples:
  - transaction aggregation
  - recursive block validation $\Rightarrow$ rapid sync
  - verifiable builds
  - private and verifiable machine learning

Fixed Circuit

- optimizable
  - high per-application cost
  - faster + fewer resources required
- one prover and one verifier *for each* application
- examples:
  - zk-evaluation of UTXO-to-nullifier map
  - zk-verification of no-inflation + output-positivity
  - wrapping a STARK

Motivation

Virtual Machine $\longleftrightarrow$ Fixed Circuit

Virtual Machine

- versatile
  - high one-time cost
  - low per-application cost
  - slower + more resources required
- one prover, one verifier, many programs
- examples:
  - transaction aggregation
  - recursive block validation $\Rightarrow$ rapid sync
  - verifiable builds
  - private and verifiable machine learning
- stateful
  - evolution across time

Fixed Circuit

- optimizable
  - high per-application cost
  - faster + fewer resources required
- one prover and one verifier *for each* application
- examples:
  - zk-evaluation of UTXO-to-nullifier map
  - zk-verification of no-inflation + output-positivity
  - wrapping a STARK
- stateless
  - direct relation between input and output

Motivation

Virtual Machine $\longleftrightarrow$ Fixed Circuit

Virtual Machine

- versatile
  - high one-time cost
  - low per-application cost
  - slower + more resources required
- one prover, one verifier, many programs
- examples:
  - transaction aggregation
  - recursive block validation $\Rightarrow$ rapid sync
  - verifiable builds
  - private and verifiable machine learning
- stateful
  - evolution across time

Fixed Circuit

- optimizable
  - high per-application cost
  - faster + fewer resources required
- one prover and one verifier *for each* application
- examples:
  - zk-evaluation of UTXO-to-nullifier map
  - zk-verification of no-inflation + output-positivity
  - wrapping a STARK
- stateless
  - direct relation between input and output

STARKs are tailored towards proving the integral evolution of a state across time.

zk-VMs that use STARKs

			open source				
zkVM	**ISA**	**team**	**verifier**	**GPU prover**	**mainnet capable**	**Ethproofs**	**based on STARK?**
Pico	RISC-V	Brevis	✓✓ dual	ETA: soon™	✓	✓	✓
SP1	RISC-V	Succinct	✓✓ dual	binaries only	✓	✓	✓
Ziren	MIPS	ZKM	✓✓ dual	closed	✓	✓	✓
ZisK	RISC-V	ZisK	✓✓ dual	✓✓ dual	✓	✓	✓
Airbender	RISC-V	Matter Labs	✓✓ dual	✓✓ dual	✓	ETA: soon™	
Ceno	RISC-V	Scroll	✓✓ dual	ETA: soon™	✓	ETA: soon™	
OpenVM	RISC-V	Axiom	✓✓ dual	✓✓ dual	✓	ETA: soon™	✓
→ Euclid	RISC-V	Scroll	✓✓ dual	closed	✓	ETA: soon™	✓
→ powdr	RISC-V	powdr	✓✓ dual	N/A	✓	ETA: soon™	✓
R0VM	RISC-V	RISC Zero	✓ Apache 2.0	✓ Apache 2.0	✓	ETA: soon™	✓
Ix	Lean 4	Argument	✓✓ dual	N/A	ETA: 2025 (no recursion)	ETA: 2025	✓
Jolt	RISC-V	a16z	✓✓ dual	ETA: soon™	ETA: 2025 (no streaming)	ETA: 2025	
Ligetron	WASM	Ligero	✓ Apache 2.0	✓ Apache 2.0	ETA: 2025 (no recursion)	ETA: 2025	
Linea EVM	EVM	Linea	✓✓ dual	closed	ETA: 2025 (no MPT)	ETA: 2025	
Miden VM	Miden ISA	Miden	✓✓ dual	✓ MIT	ETA: 2025 (no recursion)	ETA: 2025	✓
o1VM	RISC-V	O(1) Labs	✓✓ dual	N/A	ETA: 2025 (no recursion)	ETA: 2025	
Valida VM	Valida ISA	Lita	✓✓ dual	N/A	ETA: 2025 (no recursion)	ETA: 2025	✓
zkEngine	WASM	ICME	✓✓ dual	N/A	ETA: 2025 (no recursion)	ETA: 2025	
zkWASM	WASM	Delphinus	✓✓ dual	GPL 3.0	ETA: 2025 (64MB limit)	ETA: 2025	
mainnet capable in 2026?							
Aztec VM	Brillig ISA	Aztec	✓ Apache 2.0	N/A	ETA: 2026+	ETA: 2026+	
Cairo	Cairo ISA	StarkWare	✓✓ dual	✓ Apache 2.0	ETA: 2026+	ETA: 2026+	✓
Cairo M	Cairo ISA	Kakarot	✓✓ dual	N/A	ETA: 2026+	ETA: 2026+	✓
Keth	EVM	Kakarot	✓✓ dual	N/A	ETA: 2026+	ETA: 2026+	✓
Nock VM	Nock ISA	Zorp	✓✓ dual	N/A	ETA: 2026+	ETA: 2026+	✓
Petra VM	Petra ISA	Irreducible	✓ Apache 2.0	N/A	ETA: 2026+	ETA: 2026+	
Starstream	WASM	Paima	✓✓ dual	N/A	ETA: 2026+	ETA: 2026+	
Triton VM	Triton ISA	Neptune	✓✓ dual	N/A	ETA: 2026+	ETA: 2026+	✓
permissive soon?							
Nexus zkVM 3.0	RISC-V	Nexus	BUSL 1.1	N/A	ETA: 2025 (no recursion)	ETA: 2025	✓
SP1 Hypercube	RISC-V	Succinct	unlicensed	closed	✓	✓	
StarkV	RISC-V	StarkWare	ETA: 2025	N/A	ETA: 2025 (nascent)	ETA: 2025	✓
ZippelVM	RISC-V	Zippel	ETA: 2025	N/A	ETA: 2026+ (nascent)	ETA: 2026+	
[redacted #1]	RISC-V	[redacted]	ETA: 2025	N/A	ETA: 2025 (nascent)	ETA: 2025	
[redacted #2]	RISC-V	[redacted]	ETA: 2025	ETA: 2025	ETA: 2025	ETA: 2025	

credit: EthProofs

Table of Contents

Table of Contents

Table of Contents

Definition

Scalable, **T**ransparent **AR**gument of **K**nowledge

STARK[1]: any proof system that is

- *transparent* $\Rightarrow$ no trusted setup
- *succinct* $\Rightarrow$ $O(\text{poly}\log n)$ verifier
- *scalable* $\Rightarrow$ succinct + $O(n \log n)$ prover
- *argument* $\Rightarrow$ computationally sound
- *of knowledge* $\Rightarrow$ can extract witness*
- *interactive or non-interactive*
- *no preprocessing*

*in some unrealistic world, e.g., ROM or rewinding

Definition

Scalable, **T**ransparent **AR**gument of **K**nowledge

STARK[1]: any proof system that is

- *transparent* $\Rightarrow$ no trusted setup
- *succinct* $\Rightarrow$ $O(\text{poly} \log n)$ verifier
- *scalable* $\Rightarrow$ succinct + $O(n \log n)$ prover
- *argument* $\Rightarrow$ computationally sound
- *of knowledge* $\Rightarrow$ can extract witness*
- *interactive or non-interactive*
- *no preprocessing*

STARK[2]: a specific family of proof systems with

- *algebraic execution trace (AET)* and *algebraic intermediate representation (AIR)*
- *randomized AIR with preprocessing (RAP)*
- *ALI / DEEP-ALI*
- an *interactive oracle proof of proximity* such as *FRI / STIR / WHIR*
- *Merkle trees*

*in some unrealistic world, e.g., ROM or rewinding

Definition

Scalable, **T**ransparent **AR**gument of **K**nowledge

STARK[1]: any proof system that is

- *transparent* $\Rightarrow$ no trusted setup
- *succinct* $\Rightarrow$ $O(\mathrm{poly}\log n)$ verifier
- *scalable* $\Rightarrow$ succinct + $O(n \log n)$ prover
- *argument* $\Rightarrow$ computationally sound
- *of knowledge* $\Rightarrow$ can extract witness*
- *interactive or non-interactive*
- *no preprocessing*

STARK[2]: a specific family of proof systems with

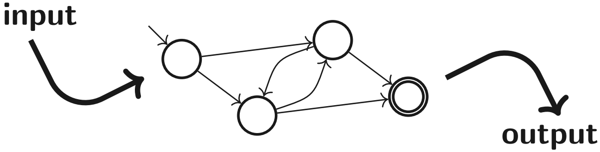
- *algebraic execution trace (AET)* and *algebraic intermediate representation (AIR)*
- *randomized AIR with preprocessing (RAP)*
- *ALI / DEEP-ALI*
- an *interactive oracle proof of proximity* such as *FRI / STIR / WHIR*
- *Merkle trees*

STARK[3]: a concrete *non-interactive* proof for some STARK[2]

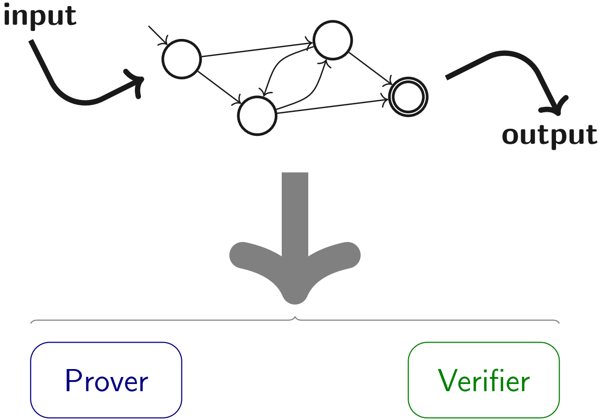
*in some unrealistic world, e.g., ROM or rewinding

Concept

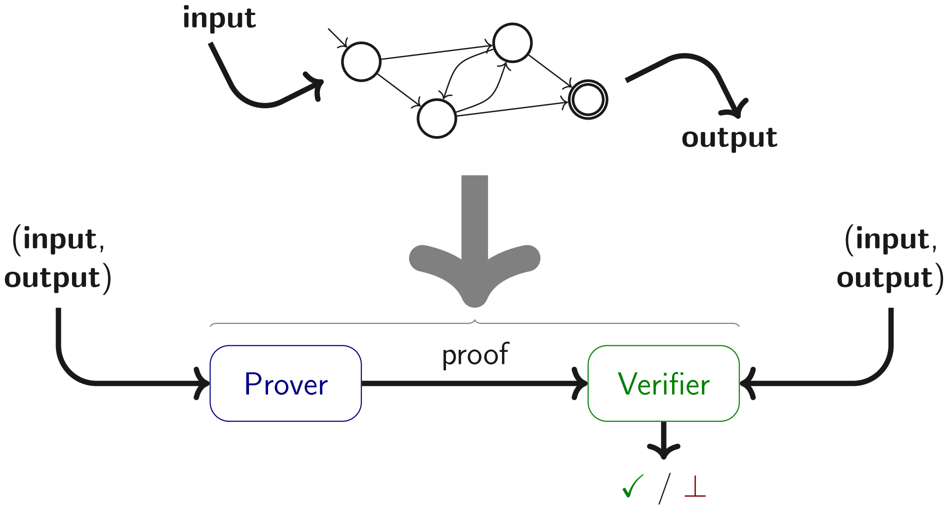
Concept



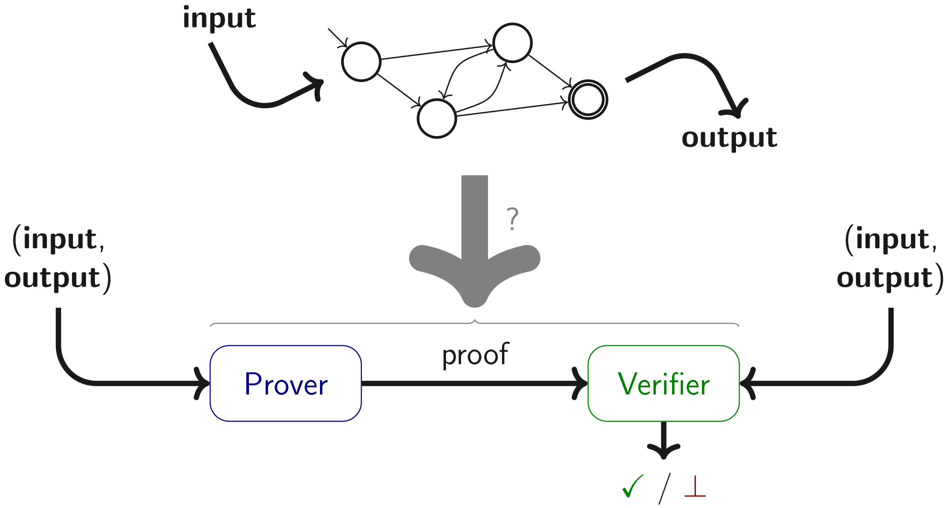
Concept



Concept

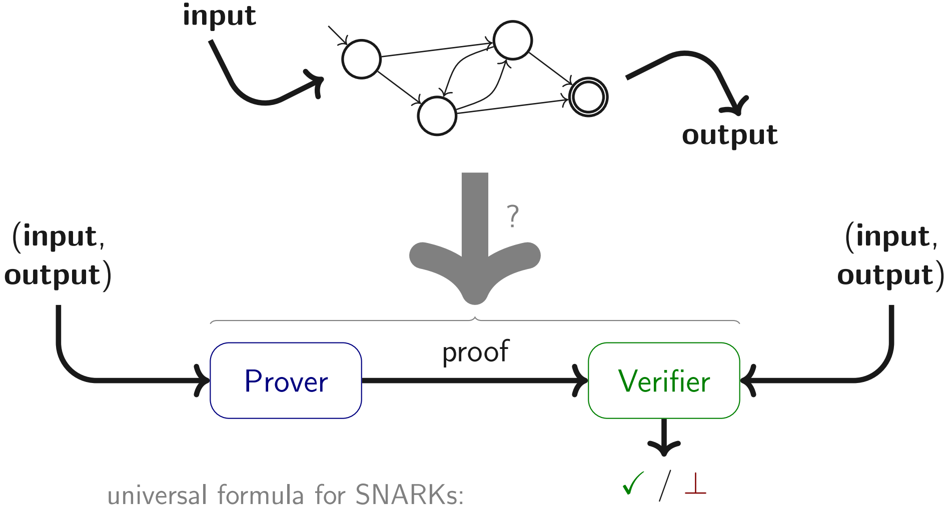

input
output
(input, output)
(input, output)
proof
Prover
Verifier
✓ / ⊥

Concept



input
output
(input, output)
?
proof
Prover
Verifier
(input, output)
✓ / ⊥

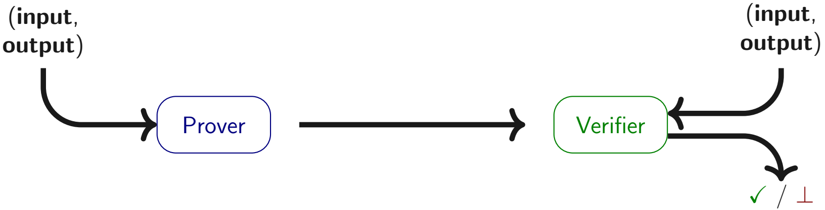
Concept



universal formula for SNARKs:

1. give Verifier superpowers
2. strip them away

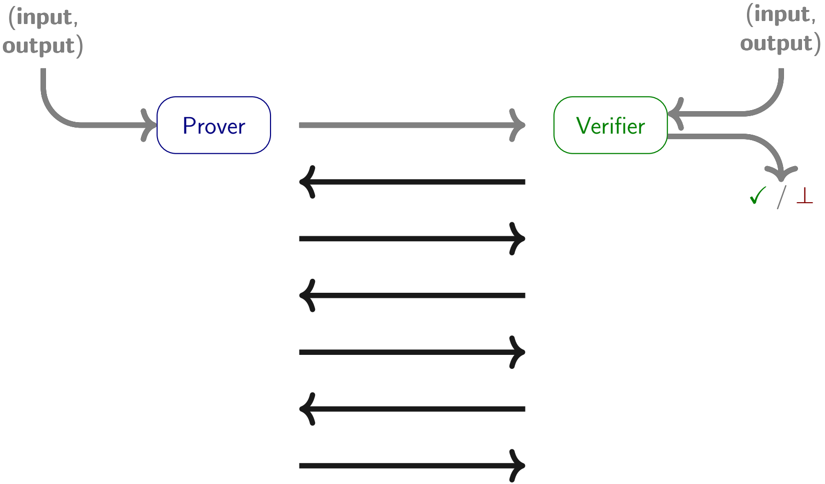
Superpowers



Superpowers

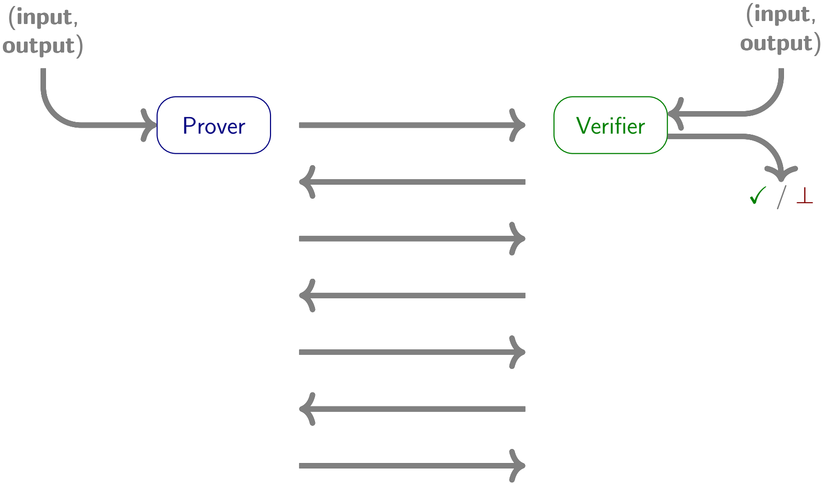
Superpowers

1. interaction



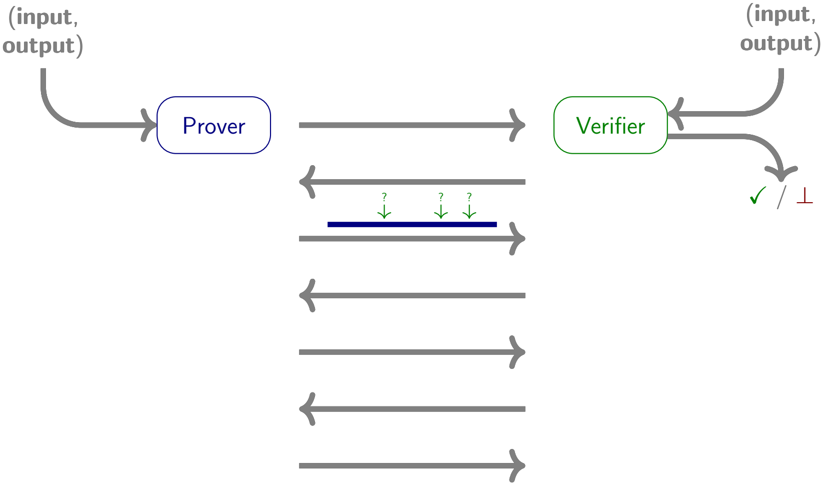
Superpowers

1. interaction



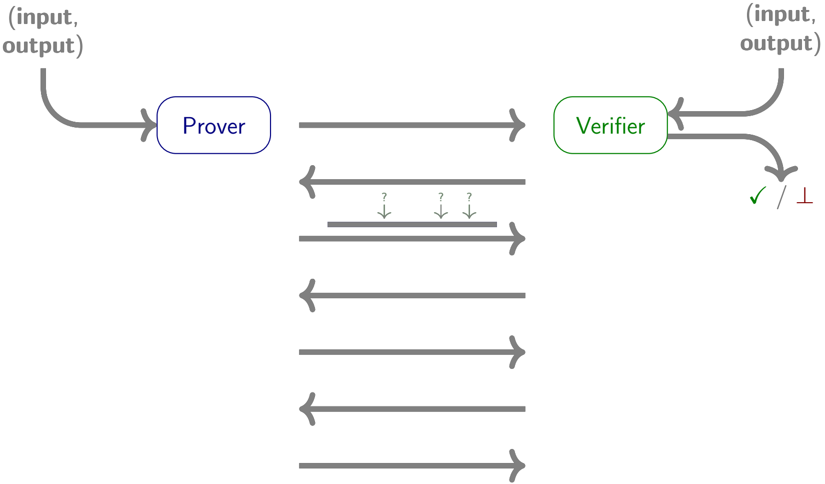
Superpowers

1. interaction
2. point queries



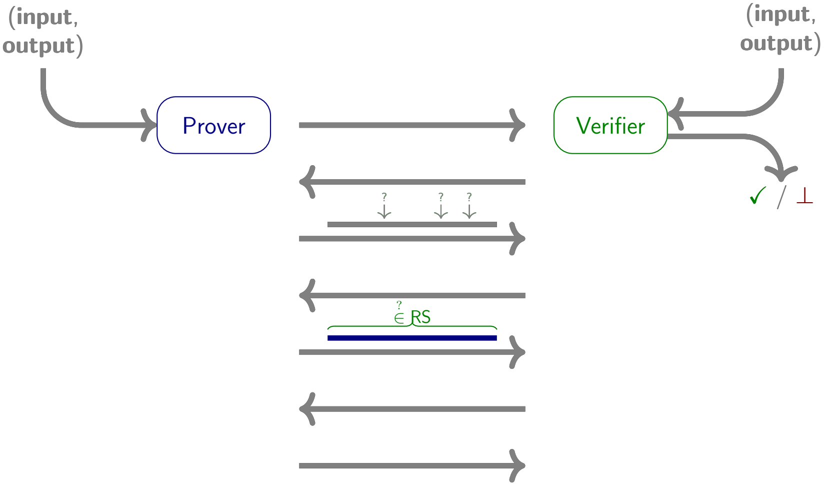
Superpowers

1. interaction
2. point queries



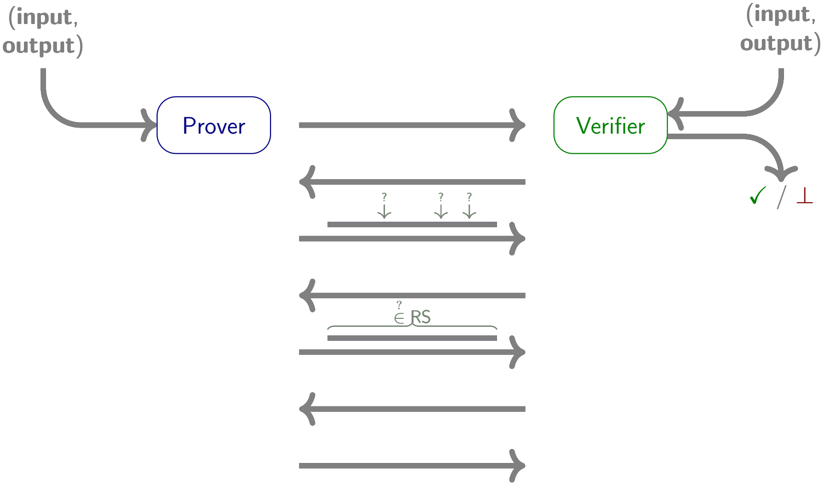
Superpowers

1. interaction
2. point queries
3. low-degree tests



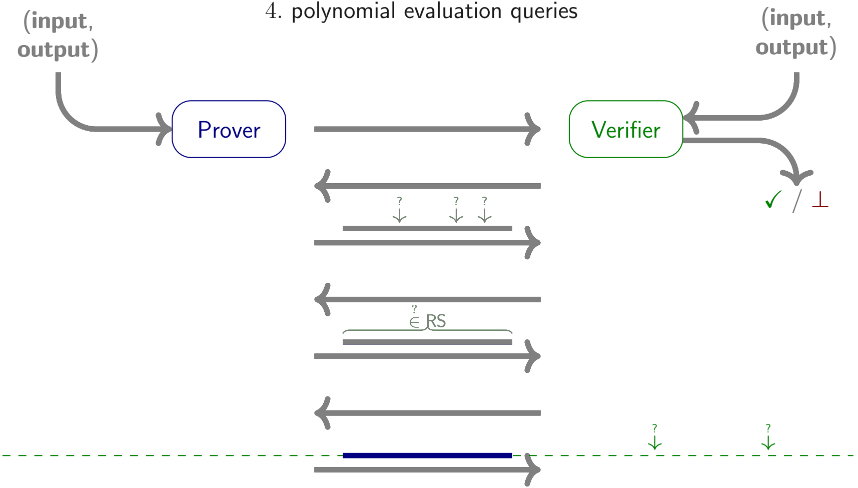
Superpowers

1. interaction
2. point queries
3. low-degree tests



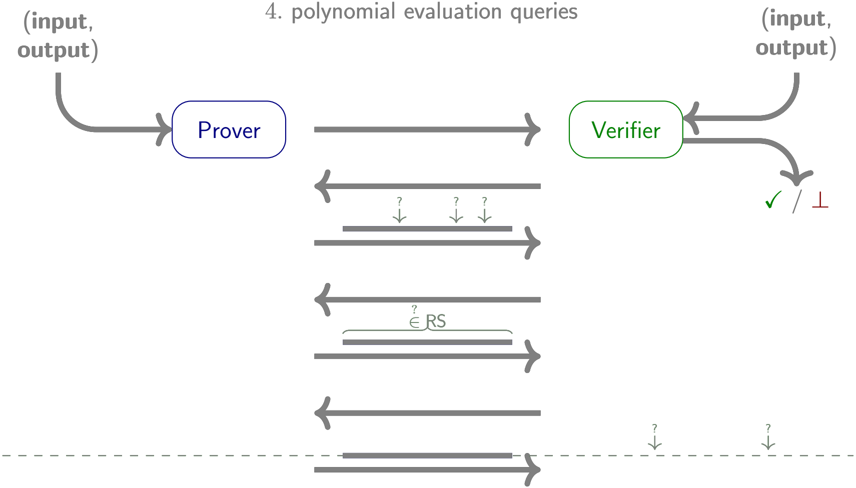
Superpowers

1. interaction
2. point queries
3. low-degree tests
4. polynomial evaluation queries



Superpowers

1. interaction
2. point queries
3. low-degree tests
4. polynomial evaluation queries

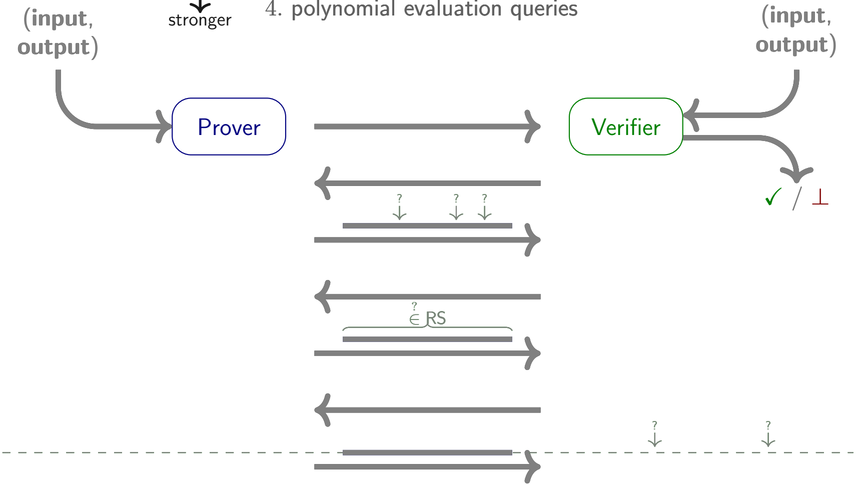


Superpowers

weaker

stronger

1. interaction
2. point queries
3. low-degree tests
4. polynomial evaluation queries



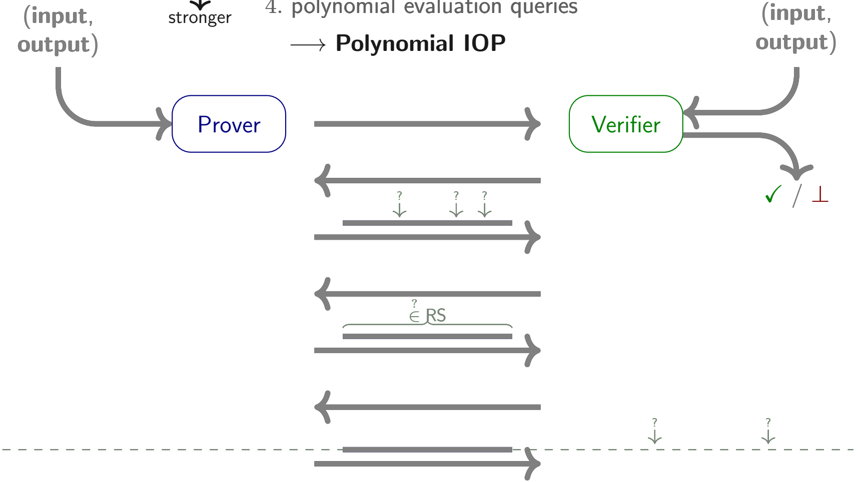
Superpowers

weaker

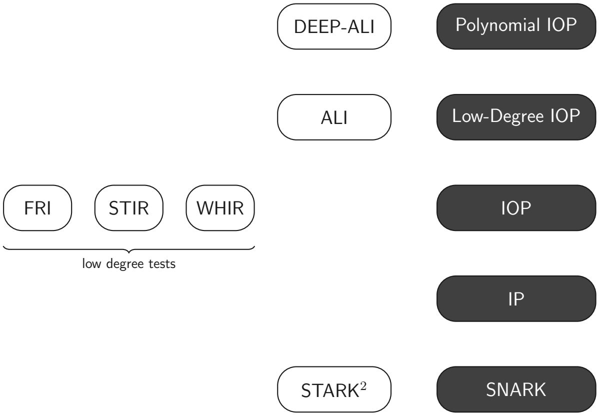
stronger

1. interaction
2. point queries $\longrightarrow$ **IOP** (interactive oracle proof)
3. low-degree tests
4. polynomial evaluation queries

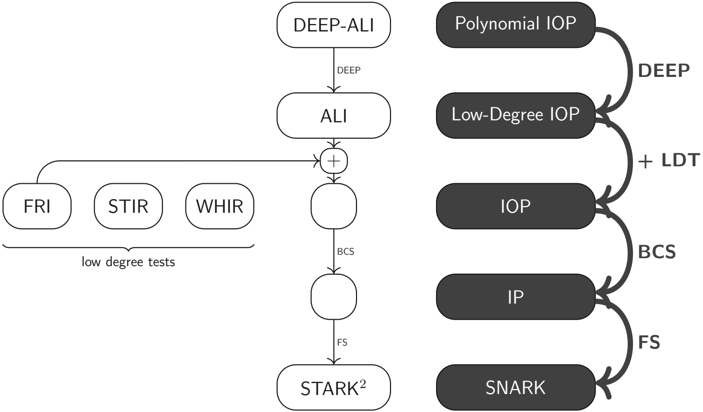
$\longrightarrow$ **Polynomial IOP**



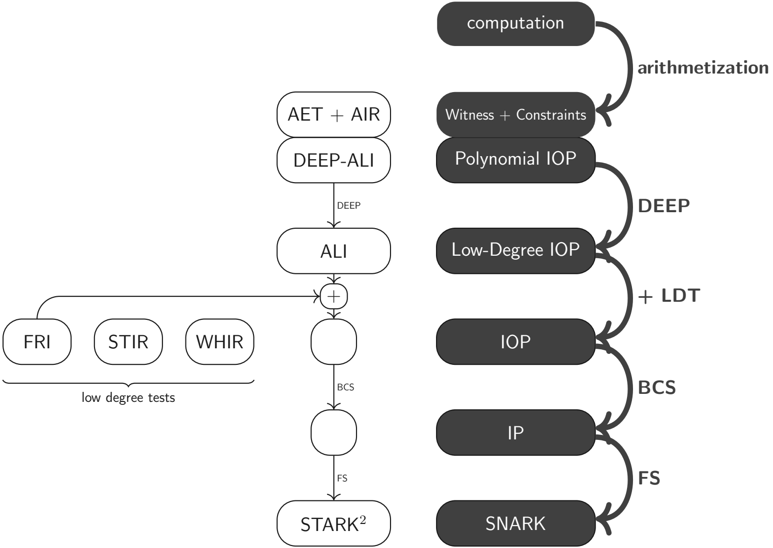
STARK Compilation Pipeline



STARK Compilation Pipeline



STARK Compilation Pipeline



STARK Compilation Pipeline

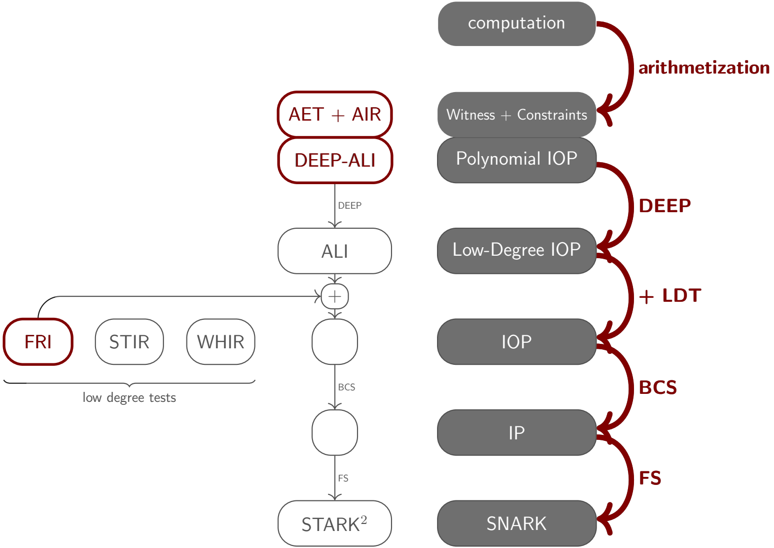
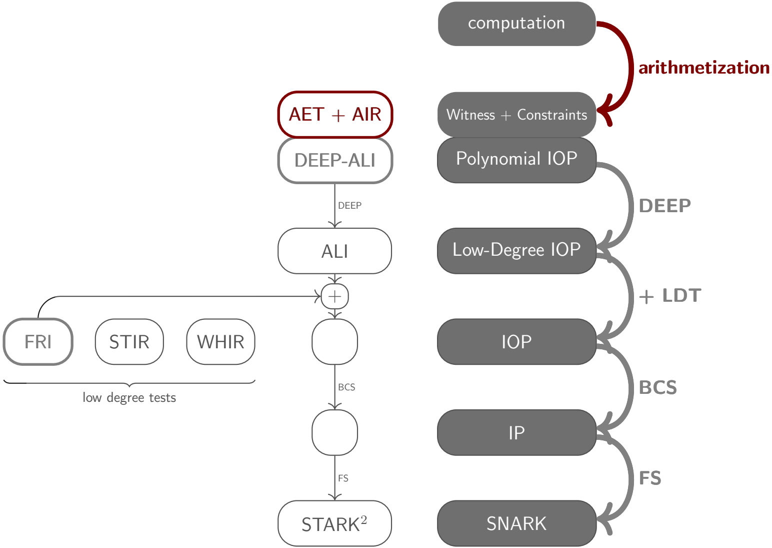


Table of Contents

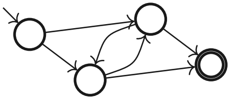
Table of Contents

STARK Compilation Pipeline (Arithmetization)

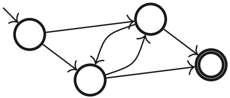


State Machine Arithmetization

State Machine Arithmetization



State Machine Arithmetization

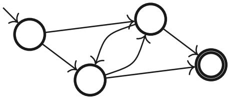


arithmetize:

describe in terms of

- finite field elements
- low degree polynomials over finite fields

State Machine Arithmetization



arithmetize:

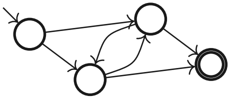
describe in terms of

– finite field elements

– low degree polynomials over finite fields

state space: $\mathbb{F}^{\mathrm{w}}$ *vectors of* w *finite field elements*

State Machine Arithmetization



arithmetize:

describe in terms of

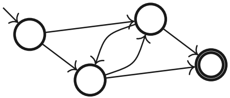
– finite field elements

– low degree polynomials over finite fields

state space: $\mathbb{F}^{\mathsf{w}}$ *vectors of* w *finite field elements*

→ *e.g.* initial state $(x_0, y_0) \in \mathbb{F}^2$

State Machine Arithmetization



arithmetize:

describe in terms of

– finite field elements

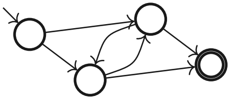
– low degree polynomials over finite fields

state space: $\mathbb{F}^{\mathsf{w}}$ *vectors of* w *finite field elements*

$\rightarrow$ *e.g.* initial state $(x_0, y_0) \in \mathbb{F}^2$

transition function: $F : \mathbb{F}^{\mathsf{w}} \rightarrow \mathbb{F}^{\mathsf{w}}$ and $F(\boldsymbol{x}) \in (\mathbb{F}[\boldsymbol{x}]^{\leqslant d})^{\mathsf{w}}$

State Machine Arithmetization



arithmetize:

describe in terms of

– finite field elements

– low degree polynomials over finite fields

state space: $\mathbb{F}^{\mathsf{w}}$ *vectors of* w *finite field elements*

→ *e.g.* initial state $(x_0, y_0) \in \mathbb{F}^2$

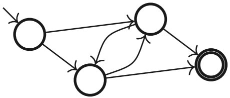
transition function: $F : \mathbb{F}^{\mathsf{w}} \to \mathbb{F}^{\mathsf{w}}$ and $F(\boldsymbol{x}) \in (\mathbb{F}[\boldsymbol{x}]^{\leqslant d})^{\mathsf{w}}$

→ evolution $\boldsymbol{x}_0 \xrightarrow{F} \boldsymbol{x}_1 \xrightarrow{F} \boldsymbol{x}_2 \xrightarrow{F} \cdots$

→ *e.g.*

$$F(x, y) = \begin{pmatrix} 1 + xy \\ x^2 + y^2 \end{pmatrix}$$

State Machine Arithmetization



arithmetize:

describe in terms of

– finite field elements

– low degree polynomials over finite fields

state space: $\mathbb{F}^{\mathsf{w}}$ *vectors of* w *finite field elements*

$\rightarrow$ *e.g.* initial state $(x_0, y_0) \in \mathbb{F}^2$

transition function: $F : \mathbb{F}^{\mathsf{w}} \rightarrow \mathbb{F}^{\mathsf{w}}$ and $F(\boldsymbol{x}) \in (\mathbb{F}[\boldsymbol{x}]^{\leqslant d})^{\mathsf{w}}$

$\rightarrow$ evolution $\boldsymbol{x}_0 \xrightarrow{F} \boldsymbol{x}_1 \xrightarrow{F} \boldsymbol{x}_2 \xrightarrow{F} \cdots$

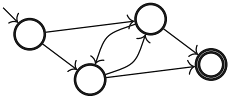
$\rightarrow$ *e.g.*

$$F(x, y) = \begin{pmatrix} 1 + xy \\ x^2 + y^2 \end{pmatrix}$$

computational claim: *boundary constraints*

$\rightarrow$ *e.g.* $\{x_0 = 1, y_0 = 2, x_7 = 3\}$

State Machine Arithmetization



arithmetize:

describe in terms of

– finite field elements

– low degree polynomials over finite fields

state space: $\mathbb{F}^{\mathsf{w}}$ *vectors of* w *finite field elements*

$\rightarrow$ *e.g.* initial state $(x_0, y_0) \in \mathbb{F}^2$

transition function: $F : \mathbb{F}^{\mathsf{w}} \rightarrow \mathbb{F}^{\mathsf{w}}$ and $F(\boldsymbol{x}) \in (\mathbb{F}[\boldsymbol{x}]^{\leqslant d})^{\mathsf{w}}$

$\rightarrow$ evolution $\boldsymbol{x}_0 \xrightarrow{F} \boldsymbol{x}_1 \xrightarrow{F} \boldsymbol{x}_2 \xrightarrow{F} \cdots$

$\rightarrow$ *e.g.*

$$F(x, y) = \begin{pmatrix} 1 + xy \\ x^2 + y^2 \end{pmatrix}$$

computational claim: *boundary constraints*

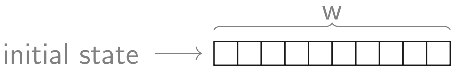
$\rightarrow$ *e.g.* $\{x_0 = 1, y_0 = 2, x_7 = 3\}$

integrity:

1. *boundary constraints* ✓
2. *transition constraints* ✓ $\quad \Leftrightarrow \quad \forall i \; \boldsymbol{x}_{i+1} = F(\boldsymbol{x}_i)$

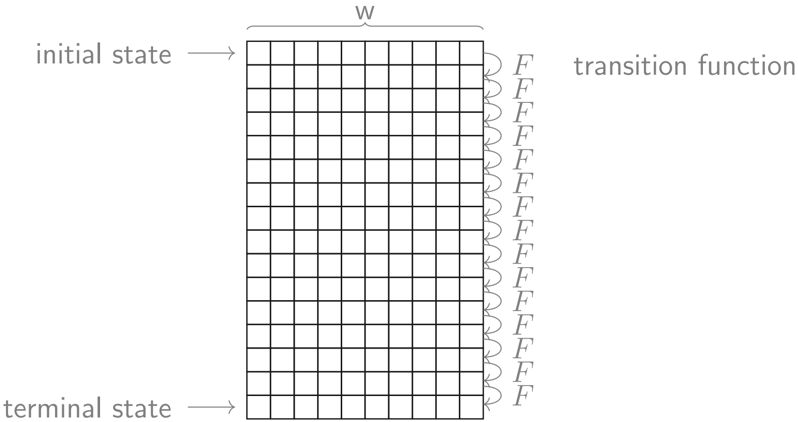
Algebraic Execution Trace (AET)

Algebraic Execution Trace (AET)



Algebraic Execution Trace (AET)

Algebraic Execution Trace (AET)



AIR

AIR:

set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AIR

AIR:

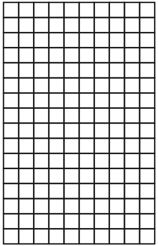
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero ⇔ AET is integral

AET



AIR constraint

type

AIR

AIR:

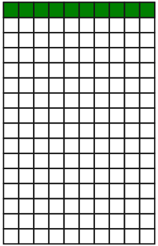
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint	type
$\boldsymbol{x}_0 - \boldsymbol{x}_{init} = 0$	initial

AIR

AIR:

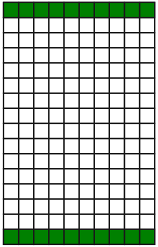
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint	type
$\boldsymbol{x}_0 - \boldsymbol{x}_{init} = 0$	initial
$a(\boldsymbol{x}_{N-1}) = 0$	terminal

AIR

AIR:

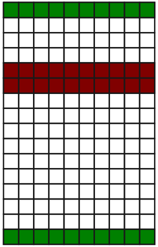
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint	type
$\boldsymbol{x}_0 - \boldsymbol{x}_{init} = 0$	initial
$\boldsymbol{x}_{i+1} - F(\boldsymbol{x}_i) = 0$	transition
$a(\boldsymbol{x}_{N-1}) = 0$	terminal

AIR

AIR:

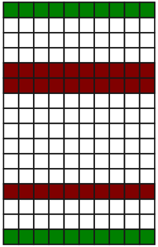
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint	type
$\boldsymbol{x}_0 - \boldsymbol{x}_{init} = 0$	initial
$\boldsymbol{x}_{i+1} - F(\boldsymbol{x}_i) = 0$	transition
$c(\boldsymbol{x}_j) = 0$	consistency
$a(\boldsymbol{x}_{N-1}) = 0$	terminal

AIR

AIR:

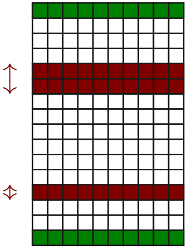
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint	type
$\boldsymbol{x}_0 - \boldsymbol{x}_{init} = 0$	initial
$\boldsymbol{x}_{i+1} - F(\boldsymbol{x}_i) = 0$	transition
$c(\boldsymbol{x}_j) = 0$	consistency
$a(\boldsymbol{x}_{N-1}) = 0$	terminal

unquantified

– applies to fixed row (combination)

quantified

– applies for all row combinations

$\forall i \in \{0, \ldots, N-2\} : \boldsymbol{x}_{i+1} - F(\boldsymbol{x}_i) = 0$

$\forall j \in \{0, \ldots, N-1\} : c(\boldsymbol{x}_j) = 0$

AIR

AIR:

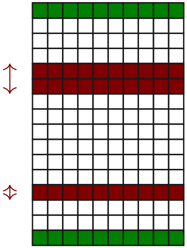
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint	type
$\boldsymbol{x}_0 - \boldsymbol{x}_{init} = 0$	initial
$\boldsymbol{x}_{i+1} - F(\boldsymbol{x}_i) = 0$	transition
$c(\boldsymbol{x}_j) = 0$	consistency
$a(\boldsymbol{x}_{N-1}) = 0$	terminal

unquantified

– applies to fixed row (combination)

quantified

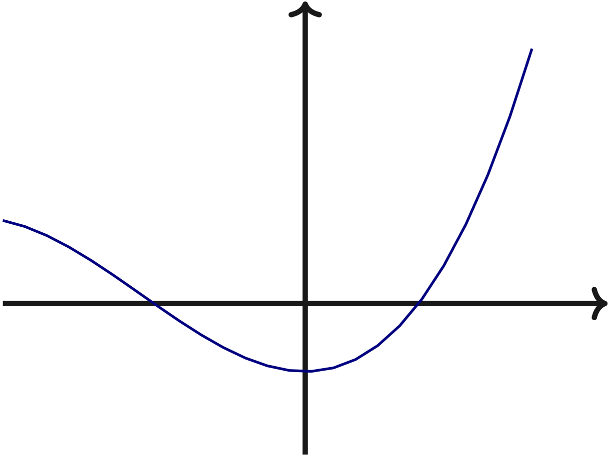
– applies for all row combinations

$\forall i \in \{0, \ldots, N-2\} : \boldsymbol{x}_{i+1} - F(\boldsymbol{x}_i) = 0$

$\forall j \in \{0, \ldots, N-1\} : c(\boldsymbol{x}_j) = 0$

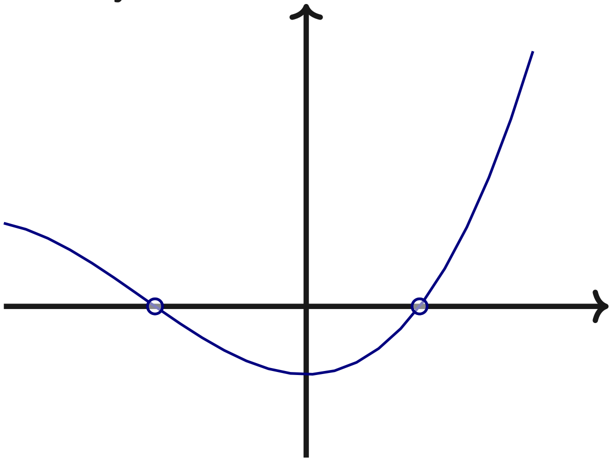
Q: How to *succinctly* test *quantified* constraints?

Divisibility



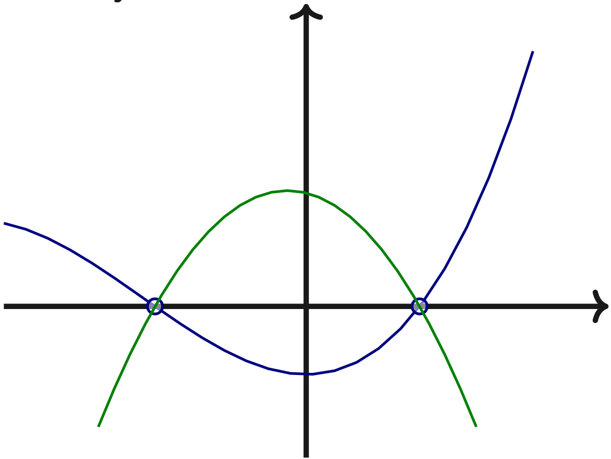
1. Given a polynomial $f(X)$

Divisibility



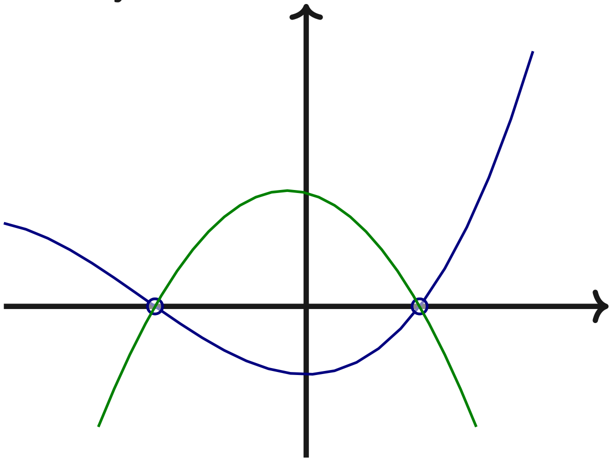
1. Given a polynomial $f(X)$ with zeros in $\{x_0, x_1, \ldots\}$

Divisibility



1. Given a polynomial $f(X)$ with zeros in $\{x_0, x_1, \ldots\}$
2. Consider the *zerofier* [2] $Z(X) = \prod_i (X - x_i)$

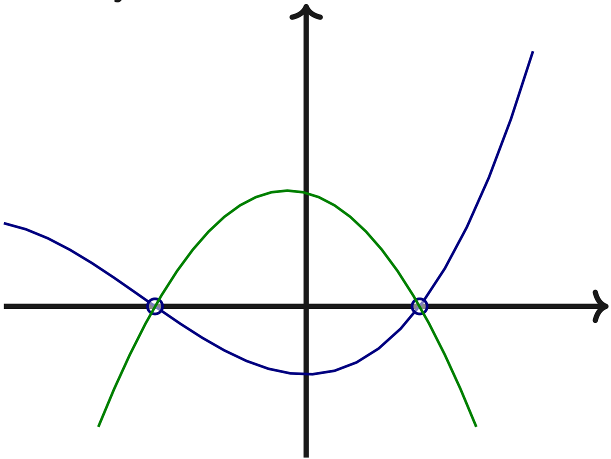
Divisibility



1. Given a polynomial $f(X)$ with zeros in $\{x_0, x_1, \dots\}$
2. Consider the *zerofier* [2] $Z(X) = \prod_i (X - x_i)$

$\therefore Z(X) \mid f(X)$

Divisibility



1. Given a polynomial $f(X)$ with zeros in $\{x_0, x_1, \ldots\}$
2. Consider the *zerofier* [2] $Z(X) = \prod_i (X - x_i)$

$\therefore Z(X) \mid f(X)$

Proof.

Let $f(X) = k(X) \cdot Z(X) + r(X)$ with $\deg(r) < \deg(Z)$.

Then $\forall i \; f(x_i) = k(x_i) \cdot Z(x_i) + r(x_i)$

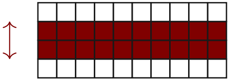
$$0 = r(x_i).$$

So $r(X)$ has more zeros than $\deg(r)$.

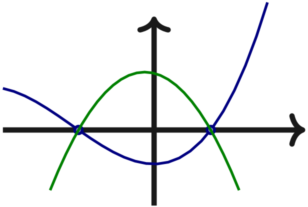
Therefore, $r(X) = 0$. $\square$

[2] also known as "vanishing polynomial"

Use Divisibility as Quantifier



$\mapsto$



quantifier

zerofier

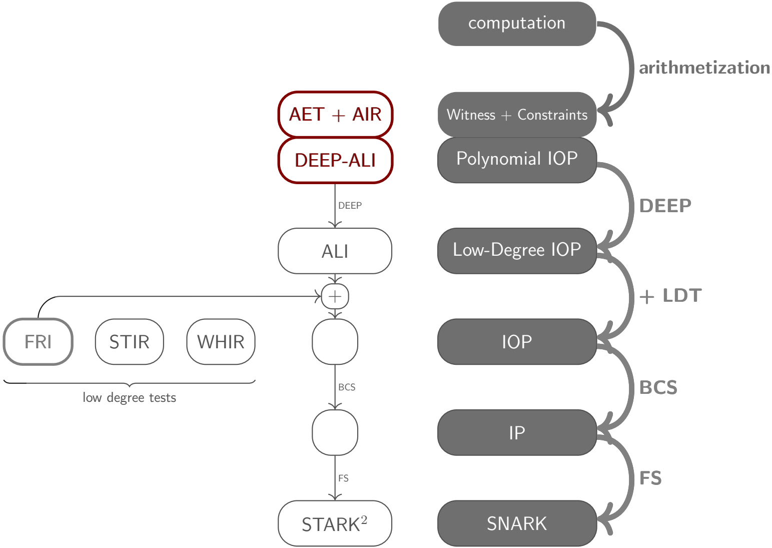
Q: how to *succinctly* test *quantified* constraints?

A: express as *polynomial divisibility* relation.

Table of Contents

Table of Contents

STARK Compilation Pipeline (DEEP-ALI)



DEEP-ALI: Intuition

1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

DEEP-ALI: Intuition

1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

Prover	Verifier
trace T	AIR $\mathcal{C}$

DEEP-ALI: Intuition

1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

Prover		Verifier
trace T		AIR $\mathcal{C}$
$\boldsymbol{t}(X) \leftarrow$ interpolate T	$\xrightarrow{\boldsymbol{t}(X)}$	

DEEP-ALI: Intuition

1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

Prover

trace T

Verifier

AIR $\mathcal{C}$

$\boldsymbol{t}(X) \leftarrow$ interpolate T

$$\xrightarrow{\quad\quad\quad\quad \boldsymbol{t}(X) \quad\quad\quad\quad}$$

for all $(c(\boldsymbol{x}_i), Z(X)) \in \mathcal{C}$ and

$$Z(X) \overset{?}{\mid} c \circ \boldsymbol{t}(X)$$

DEEP-ALI: Intuition

1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

Problems:

1. Soundness?
2. transition constraints $c(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$?
3. Polynomial IOP:

– evaluation queries ✓

– divisibility checks ×

Prover

trace T

$\boldsymbol{t}(X) \leftarrow$ interpolate T

Verifier

AIR $\mathcal{C}$

$$\xrightarrow{\boldsymbol{t}(X)}$$

for all $(c(\boldsymbol{x}_i), Z(X)) \in \mathcal{C}$ and

$$Z(X) \overset{?}{\mid} c \circ \boldsymbol{t}(X)$$

DEEP-ALI: Intuition

1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

Problems:

1. Soundness?
2. transition constraints $c(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$?
3. Polynomial IOP:

– evaluation queries ✓

– divisibility checks ×

Prover		Verifier
trace T		AIR $\mathcal{C}$
$\boldsymbol{t}(X) \leftarrow$ interpolate T	$\xrightarrow{\boldsymbol{t}(X)}$	

for all $(c(\boldsymbol{x}_i), Z(X)) \in \mathcal{C}$ and

$$Z(X) \overset{?}{\mid} c \circ \boldsymbol{t}(X)$$

Polynomial IOP for Divisibility

want:	have:
divisibility check	evaluation queries

Polynomial IOP for Divisibility

want: divisibility check		have: evaluation queries	
Prover	Verifier	Prover	Verifier
$p(X), Z(X)$	$p(X), Z(X)$	$p(X), Z(X)$	$p(X), Z(X)$

simulate

Polynomial IOP for Divisibility

want:
divisibility check

have:
evaluation queries

simulate

Prover	Verifier	Prover	Verifier
$p(X), Z(X)$	$p(X), Z(X)$	$p(X), Z(X)$	$p(X), Z(X)$
	$Z(X) \stackrel{?}{\mid} p(X)$		

Polynomial IOP for Divisibility

want:

divisibility check

have:

evaluation queries

simulate

Prover	Verifier	Prover	Verifier
$p(X), Z(X)$	$p(X), Z(X)$	$p(X), Z(X)$	$p(X), Z(X)$
	$Z(X) \overset{?}{\mid} p(X)$	$q(X) \leftarrow \frac{p(X)}{Z(X)}$	
		$\xrightarrow{q(X)}$	
			$z \overset{\$}{\leftarrow} \mathbb{F}$
			$Z(z) \cdot q(z) \overset{?}{=} p(z)$

Polynomial IOP for Divisibility

want:
divisibility check

have:
evaluation queries

simulate

Prover	Verifier
$p(X), Z(X)$	$p(X), Z(X)$
	$Z(X) \overset{?}{\mid} p(X)$

Prover		Verifier
$p(X), Z(X)$		$p(X), Z(X)$
$q(X) \leftarrow \frac{p(X)}{Z(X)}$	$\xrightarrow{q(X)}$	
		$z \xleftarrow{\$} \mathbb{F}$
		$Z(z) \cdot q(z) \overset{?}{=} p(z)$

Soundness.

$$\Pr[Z(X) \cdot q(X) \neq p(X) \wedge Z(z) \cdot q(z) = p(z)]$$

$\leqslant \frac{\deg}{|\mathbb{F}|}$ (Schwartz-Zippel)

$= \varepsilon.$

Polynomial IOP for Divisibility

want:
divisibility check

have:
evaluation queries

simulate

Prover	Verifier
$p(X), Z(X)$	$p(X), Z(X)$
	$Z(X) \overset{?}{\mid} p(X)$

Prover		Verifier
$p(X), Z(X)$		$p(X), Z(X)$
$q(X) \leftarrow \frac{p(X)}{Z(X)}$	$\xrightarrow{q(X)}$	
		$z \overset{\$}{\leftarrow} \mathbb{F}$
		$\underline{Z(z)} \cdot q(z) \overset{?}{=} p(z)$

evaluate efficiently?
$\rightarrow$ sparsity

Soundness.

$$\Pr[Z(X) \cdot q(X) \neq p(X) \wedge Z(z) \cdot q(z) = p(z)]$$
$$\leqslant \frac{\deg}{|\mathbb{F}|} \quad \text{(Schwartz-Zippel)}$$
$$= \varepsilon.$$

DEEP-ALI: Refinement

1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

Prover

trace T, AIR $\mathcal{C}$

Verifier

AIR $\mathcal{C}$

$\boldsymbol{t}(X) \leftarrow$ interpolate T

$\xrightarrow{\boldsymbol{t}(X)}$

for all $(c(\boldsymbol{x}_i), Z(X)) \in \mathcal{C}$:

$q_j(X) \leftarrow \frac{c \circ \boldsymbol{t}(X)}{Z(X)}$

$\xrightarrow{\boldsymbol{q}(X)}$

$z \xleftarrow{\$} \mathbb{F}$

for all $(c(\boldsymbol{x}_i), Z(X)) \in \mathcal{C}$:

$$q_i(z) \cdot Z(z) \stackrel{?}{=} c \circ \boldsymbol{t}(z)$$

DEEP-ALI: Refinement

1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

Problems:

~~1. Soundness?~~ $\varepsilon = \frac{\deg}{|\mathbb{F}|}$ $(1.a)$ what is $\deg$?)

2. transition constraints $c(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$?

~~3. Polynomial IOP:~~

~~– evaluation queries ✓~~

~~– divisibility checks ×~~

4. sparse zerofier

Prover		Verifier
trace T, AIR $\mathcal{C}$		AIR $\mathcal{C}$
$\boldsymbol{t}(X) \leftarrow$ interpolate T	$\xrightarrow{\boldsymbol{t}(X)}$	
for all $(c(\boldsymbol{x}_i), Z(X)) \in \mathcal{C}$:		
$q_j(X) \leftarrow \frac{c \circ \boldsymbol{t}(X)}{Z(X)}$	$\xrightarrow{\boldsymbol{q}(X)}$	
		$z \xleftarrow{\$} \mathbb{F}$
		for all $(c(\boldsymbol{x}_i), Z(X)) \in \mathcal{C}$:
		$q_i(z) \cdot Z(z) \stackrel{?}{=} c \circ \boldsymbol{t}(z)$

DEEP-ALI: Refinement

1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

Problems:

~~1. Soundness?~~ $\varepsilon = \frac{\deg}{|\mathbb{F}|}$ $(1.a)$ what is $\deg$?)

2. transition constraints $c(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$?

~~3. Polynomial IOP:~~

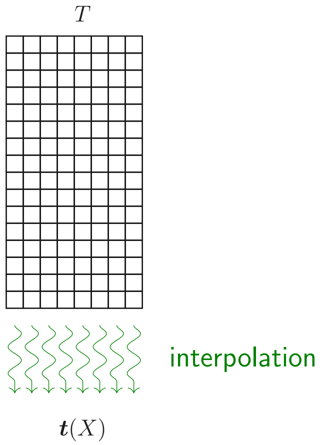
~~– evaluation queries ✓~~

~~– divisibility checks ×~~

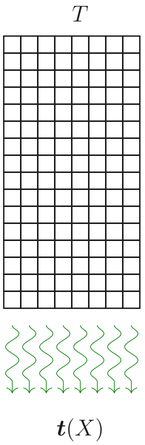
4. sparse zerofier

Prover		Verifier
trace T, AIR $\mathcal{C}$		AIR $\mathcal{C}$
$\boldsymbol{t}(X) \leftarrow$ interpolate T	$\xrightarrow{\boldsymbol{t}(X)}$	
for all $(c(\boldsymbol{x}_i), Z(X)) \in \mathcal{C}$:		
$q_j(X) \leftarrow \frac{c \circ \boldsymbol{t}(X)}{Z(X)}$	$\xrightarrow{\boldsymbol{q}(X)}$	
		$z \xleftarrow{\$} \mathbb{F}$
		for all $(c(\boldsymbol{x}_i), Z(X)) \in \mathcal{C}$:
		$q_i(z) \cdot Z(z) \stackrel{?}{=} c \circ \boldsymbol{t}(z)$

Trace Interpolation Domain



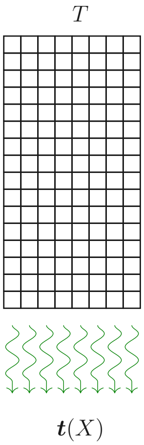
Trace Interpolation Domain



interpolation

Q: over which domain?

Trace Interpolation Domain

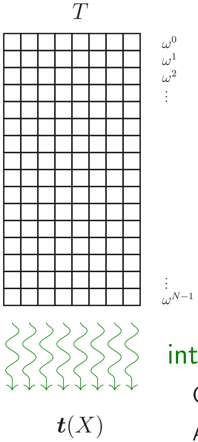


interpolation

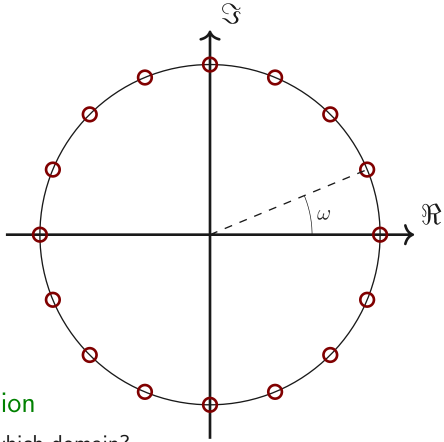
Q: over which domain?

A: subgroup of $\mathbb{F}^*$ of order $N = 2^k$

Trace Interpolation Domain



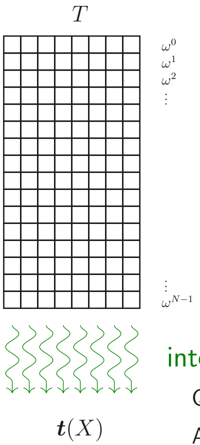
interpolation



Q: over which domain?

A: subgroup of $\mathbb{F}^*$ of order $N = 2^k$

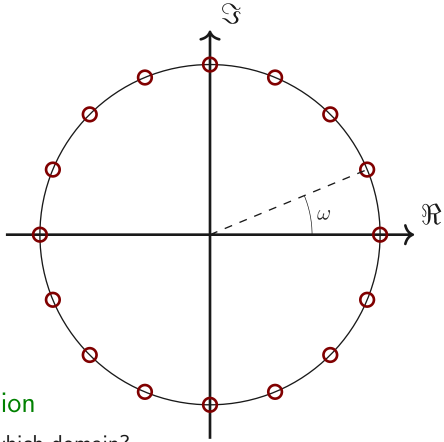
Trace Interpolation Domain



interpolation

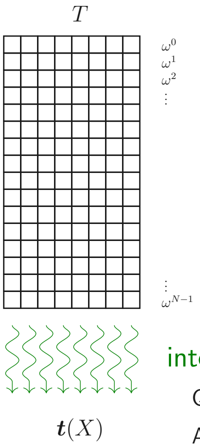
Q: over which domain?

A: subgroup of $\mathbb{F}^*$ of order $N = 2^k$



- complexity:
  $O(N \log N)$ (NTT)

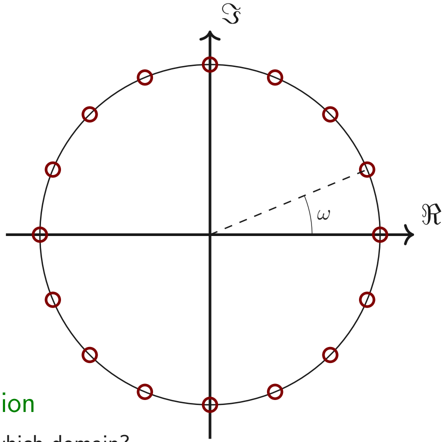
Trace Interpolation Domain



interpolation

Q: over which domain?

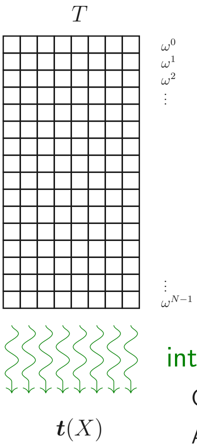
A: subgroup of $\mathbb{F}^*$ of order $N = 2^k$



- $\star$ complexity:

  $O(N \log N)$ (NTT)
- $\star$ zerofiers:
  - first row: $X - 1$
  - last row: $X - \omega^{-1}$
  - entire domain: $X^N - 1$
  - … except for last row: $\frac{X^N - 1}{X - \omega^{-1}}$

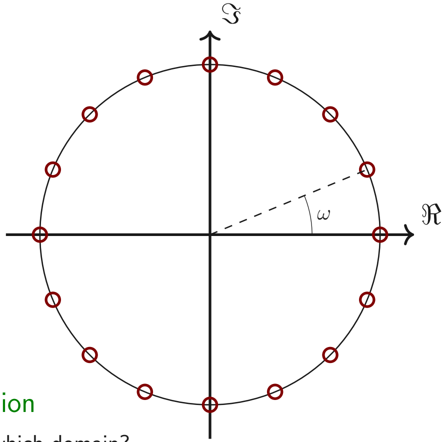
Trace Interpolation Domain



interpolation

Q: over which domain?

A: subgroup of $\mathbb{F}^*$ of order $N = 2^k$



- ⋆ complexity:
  $O(N \log N)$ (NTT)
- ⋆ zerofiers:
  - first row: $X - 1$
  - last row: $X - \omega^{-1}$
  - entire domain: $X^N - 1$
  - ... except for last row: $\frac{X^N - 1}{X - \omega^{-1}}$
- ⋆ arithmetic shift
  $\boldsymbol{t}(\omega X)$ — rotation by 1 row
  $(\boldsymbol{t}(X), \boldsymbol{t}(\omega X))$ — consecutive pairs
  $c(\boldsymbol{x}_i, \boldsymbol{x}_{i+1}) \circ \boldsymbol{t}(X) \triangleq c(\boldsymbol{t}(X), \boldsymbol{t}(\omega X))$

DEEP-ALI

1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

Problems:

~~1. Soundness?~~ $\varepsilon = \frac{\deg}{|\mathbb{F}|}$ $(1.a)$ what is $\deg$?)

~~2. transition constraints $c(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$?~~

~~3. Polynomial IOP:~~

~~– evaluation queries ✓~~

~~– divisibility checks ×~~

~~4. sparse zerofier~~

Prover

trace T, AIR $\mathcal{C}$

Verifier

AIR $\mathcal{C}$

$\boldsymbol{t}(X) \leftarrow$ interpolate T

$$\xrightarrow{\boldsymbol{t}(X)}$$

for all $(c, Z(X)) \in \mathcal{C}$:

$q_j(X) \leftarrow \frac{c \circ \boldsymbol{t}(X)}{Z(X)}$

$$\xrightarrow{\boldsymbol{q}(X)}$$

$z \xleftarrow{\$} \mathbb{F}$

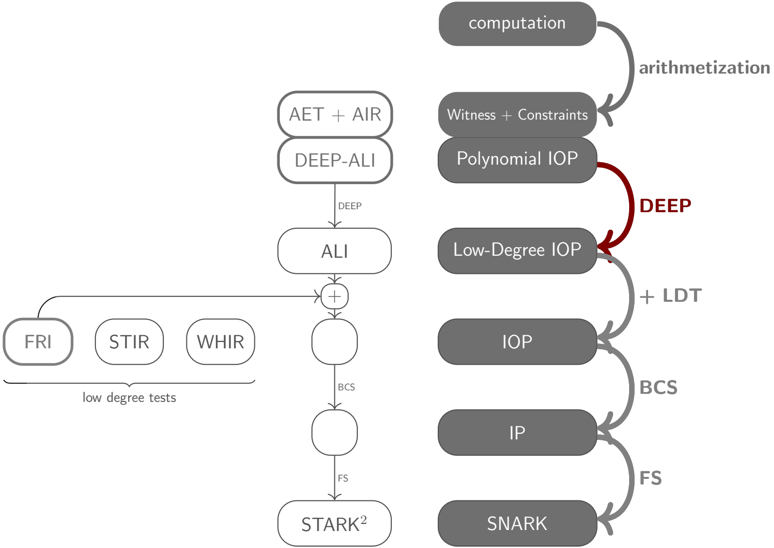
for all $(c, Z(X)) \in \mathcal{C}$:

$$q_i(z) \cdot Z(z) \stackrel{?}{=} c \circ \boldsymbol{t}(z)$$

Table of Contents

Table of Contents

STARK Compilation Pipeline (DEEP)



Polynomial IOP to Low-Degree IOP (1)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

Polynomial IOP to Low-Degree IOP (1)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover $p(X)$

Verifier $p(X)$

Prover $\boldsymbol{p}$

Verifier $\boldsymbol{p}$

Polynomial IOP to Low-Degree IOP (1)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover	Verifier	Prover	Verifier
$p(X)$	$p(X)$	$\boldsymbol{p}$	$\boldsymbol{p}$
	$z \xleftarrow{\$} \mathbb{F}$		
	$y \leftarrow p(z)$		

Reed-Solomon Code

$$\begin{array}{ccc} \text{polynomial} & \mapsto & \text{codeword} \\ p(X) & & \boldsymbol{p} \end{array}$$

Reed-Solomon Code

polynomial $\mapsto$ codeword

$p(X)$ $\qquad$ $\boldsymbol{p}$

$$\boldsymbol{p} = p(D)$$

Reed-Solomon Code in STARKs

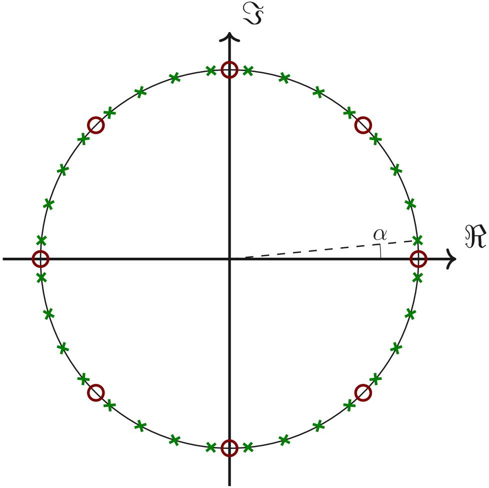
polynomial $\mapsto$ codeword

$p(X)$ $\boldsymbol{p}$

$$\boldsymbol{p} = p(D)$$

○ trace *interpolation* domain

× trace *evaluation* domain

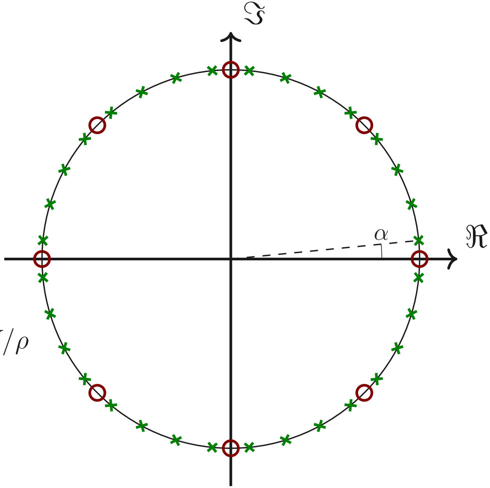


Reed-Solomon Code in STARKs

polynomial $\mapsto$ codeword

$p(X)$ $\boldsymbol{p}$

$$\boldsymbol{p} = p(D)$$



- ○ trace *interpolation* domain
- × trace *evaluation* domain
  $=$ coset of subgroup of order N/ρ

Reed-Solomon Code in STARKs

polynomial $\mapsto$ codeword

$p(X)$ $\qquad$ $\boldsymbol{p}$

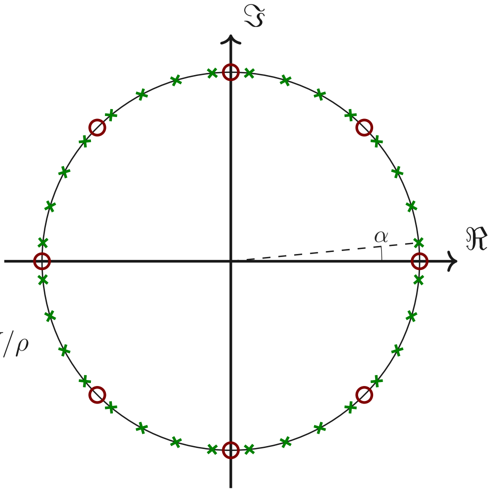
$$\boldsymbol{p} = p(D)$$

- ○ trace *interpolation* domain
- × trace *evaluation* domain

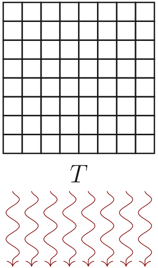
  $=$ coset of subgroup of order N/ρ

  code *rate* $\rho = \frac{\#\circ}{\#\times}$

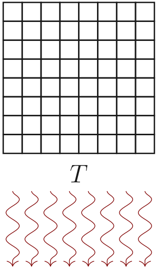
  limits $\deg$



Low-Degree Extension



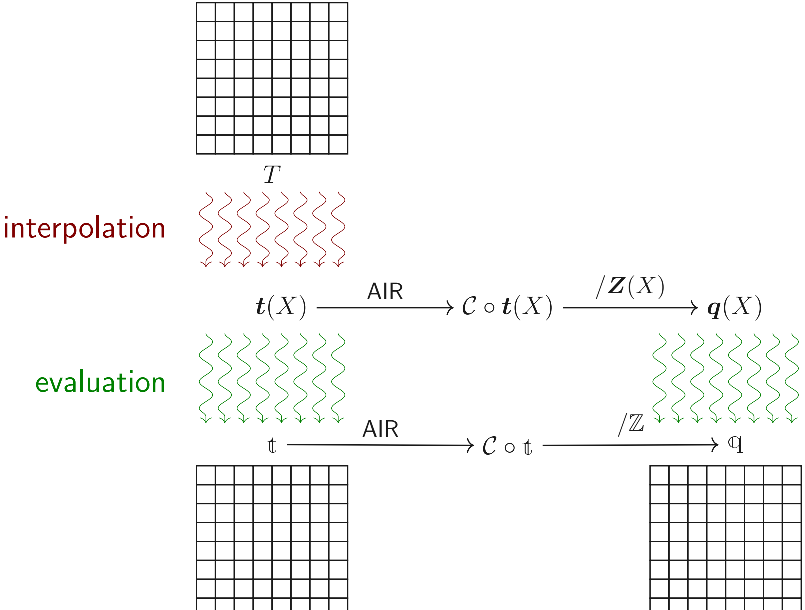
Low-Degree Extension



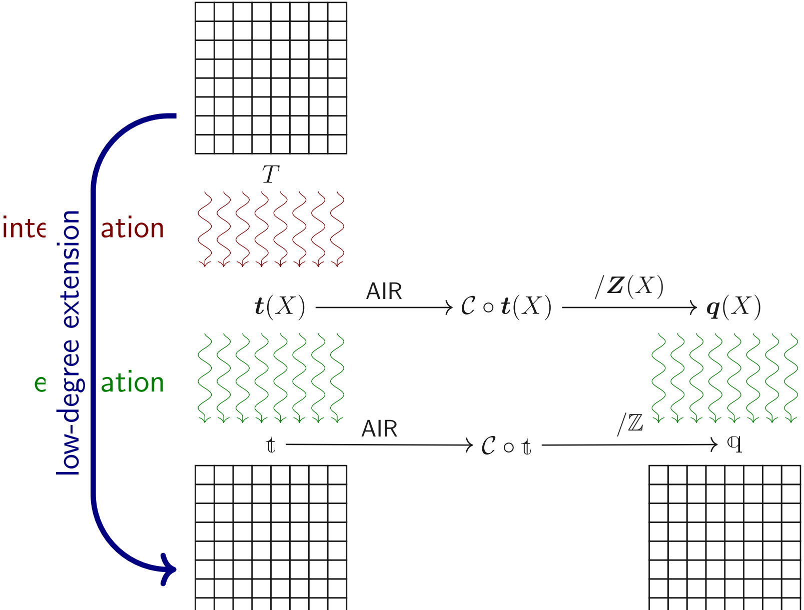
interpolation

$$\boldsymbol{t}(X) \xrightarrow{\text{AIR}} \mathcal{C} \circ \boldsymbol{t}(X) \xrightarrow{/\boldsymbol{Z}(X)} \boldsymbol{q}(X)$$

Low-Degree Extension



Low-Degree Extension



Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover	Verifier	Prover	Verifier
$p(X)$	$p(X)$	$\boldsymbol{p}$	$\boldsymbol{p}$
	$z \xleftarrow{\$} \mathbb{F}$		
	$y \leftarrow p(z)$		

Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover	Verifier
$p(X)$	$p(X)$
	$z \xleftarrow{\$} \mathbb{F}$
	$y \leftarrow p(z)$

Prover		Verifier
$\boldsymbol{p}$		$\boldsymbol{p}$
		$z \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad z \quad}$	
	$\xrightarrow{\quad p(z) \quad}$	

Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover	Verifier
$p(X)$	$p(X)$
	$z \xleftarrow{\$} \mathbb{F}$
	$y \leftarrow p(z)$

Prover		Verifier
$\boldsymbol{p}$		$\boldsymbol{p}$
		$z \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad z \quad}$	
	$\xrightarrow{\quad p(z) \quad}$	

$$\boldsymbol{q} \leftarrow \mathrm{DEEP}(\boldsymbol{p}, p(z), z) = \left[\frac{\boldsymbol{p}[i] - p(z)}{D[i] - z}\right]_i$$

Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover	Verifier
$p(X)$	$p(X)$
	$z \xleftarrow{\$} \mathbb{F}$
	$y \leftarrow p(z)$

Prover		Verifier
$\boldsymbol{p}$		$\boldsymbol{p}$
		$z \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad z \quad}$	
	$\xrightarrow{\quad p(z) \quad}$	
		$\boldsymbol{q} \leftarrow \mathsf{DEEP}(\boldsymbol{p}, p(z), z)$
		$= \left[\frac{\boldsymbol{p}[i] - p(z)}{D[i] - z}\right]_i$
		$\boldsymbol{p}, \boldsymbol{q} \overset{?}{\in} \mathsf{RS}$

Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover	Verifier
$p(X)$	$p(X)$
	$z \xleftarrow{\$} \mathbb{F}$
	$y \leftarrow p(z)$

Prover		Verifier
$\boldsymbol{p}$		$\boldsymbol{p}$
		$z \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad z \quad}$	
	$\xrightarrow{\quad p(z) \quad}$	
		$\boldsymbol{q} \leftarrow \mathsf{DEEP}(\boldsymbol{p}, p(z), z)$
		$= \left[\frac{\boldsymbol{p}[i]-p(z)}{D[i]-z}\right]_i$
		$\boldsymbol{p}, \boldsymbol{q} \overset{?}{\in} \mathsf{RS}$

Soundness.

$$\Pr\left[\left[\frac{\boldsymbol{p}[i]-y}{D[i]-z}\right]_i \in \mathsf{RS} \mid y \neq p(z)\right] = 0$$

$\therefore \varepsilon = 0.$

Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover	Verifier
$p(X)$	$p(X)$
	$z \xleftarrow{\$} \mathbb{F}$
	$y \leftarrow p(z)$

Soundness.

$$\Pr\left[\left[\frac{\boldsymbol{p}[i]-y}{D[i]-z}\right]_i \in \mathsf{RS} \mid y \neq p(z)\right] = 0$$

$\therefore \varepsilon = 0.$

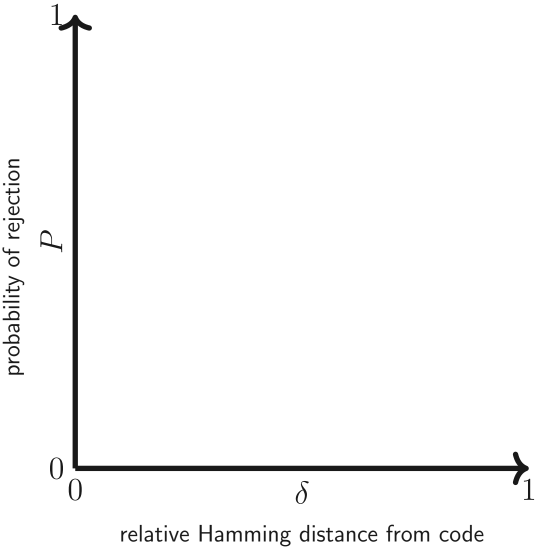
Prover		Verifier
$\boldsymbol{p}$		$\boldsymbol{p}$
		$z \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad z \quad}$	
	$\xrightarrow{\quad p(z) \quad}$	

$$\boldsymbol{q} \leftarrow \mathsf{DEEP}(\boldsymbol{p}, p(z), z)$$

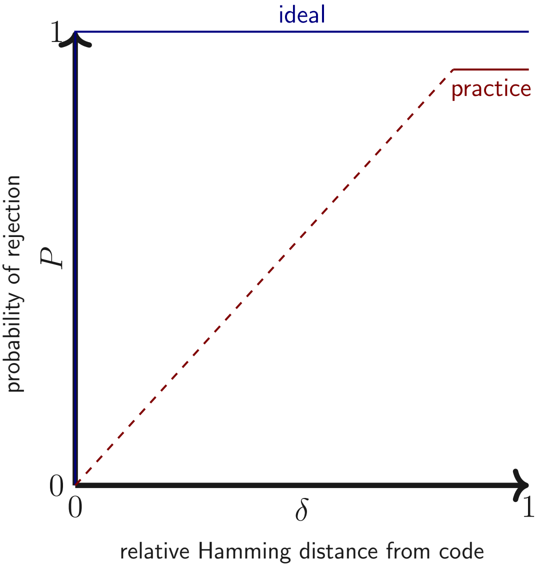
$$= \left[\frac{\boldsymbol{p}[i]-p(z)}{D[i]-z}\right]_i$$

$$\boxed{\boldsymbol{p}, \boldsymbol{q} \overset{?}{\in} \mathsf{RS}}$$

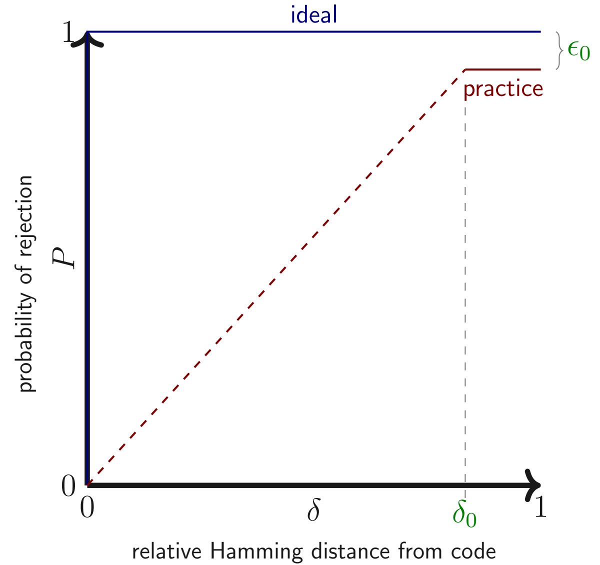
Distance Parameter



Distance Parameter



Distance Parameter



DEEP Soundness

$$\Pr\left[\Delta\left(\left[\frac{\boldsymbol{p}[i]-y}{D[i]-z}\right]_i, \mathsf{RS}\right) < \delta \,\middle|\, \forall f(X) \in \mathsf{list}(\boldsymbol{p})\,,\ f(z) \neq y\right]$$

DEEP Soundness

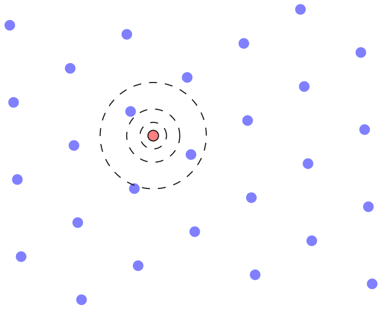
$$\Pr\left[\Delta\left(\left[\tfrac{\boldsymbol{p}[i]-y}{D[i]-z}\right]_i, \mathsf{RS}\right) < \delta \,\middle|\, \forall f(X) \in \mathsf{list}(\boldsymbol{p})\,,\ f(z) \neq y\right] \leqslant \tfrac{(\#\mathsf{list})^2}{2} \cdot \tfrac{\deg}{|\mathbb{F}|-N}$$

DEEP Soundness

$$\Pr\left[\Delta\left(\left[\frac{\boldsymbol{p}[i]-y}{D[i]-z}\right]_i, \mathsf{RS}\right) < \delta \,\middle|\, \forall f(X) \in \mathsf{list}(\boldsymbol{p})\,,\ f(z) \neq y\right] \leqslant \frac{(\#\mathsf{list})^2}{2} \cdot \frac{\deg}{|\mathbb{F}|-N}$$

$\#$list depends on δ

Reed-Solomon List Decoding Conjecture



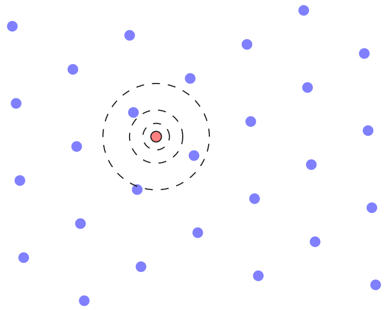
Reed-Solomon List Decoding Conjecture

decode an arbitrary point

for which radii δ

is the resulting list L

$|L| \leqslant$ poly?



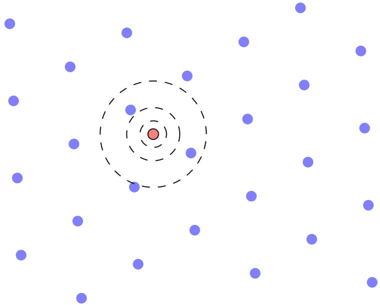
Reed-Solomon List Decoding Conjecture

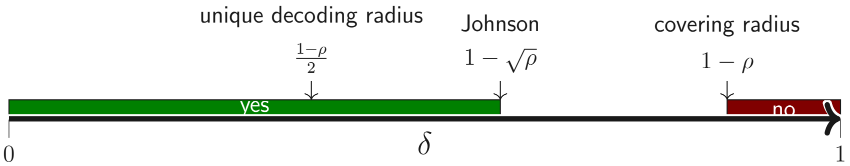
decode an arbitrary point

for which radii δ

is the resulting list L

$|L| \leqslant$ poly?





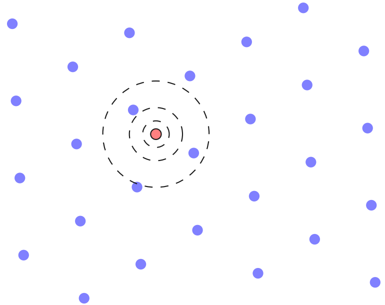
Reed-Solomon List Decoding Conjecture

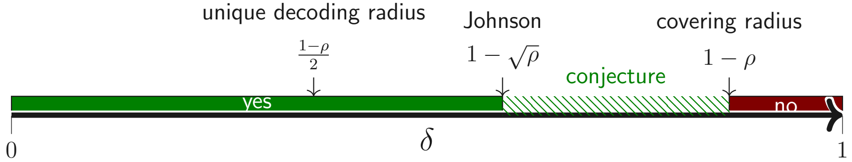
decode an arbitrary point

for which radii δ

is the resulting list L

$|L| \leqslant$ poly?





DEEP Soundness

$$\Pr\left[\Delta\left(\left[\tfrac{\boldsymbol{p}[i]-y}{D[i]-z}\right]_i, \mathsf{RS}\right) < \delta \,\middle|\, \forall f(X) \in \mathsf{list}(\boldsymbol{p}),\ f(z) \neq y\right] \leqslant \tfrac{(\#\mathsf{list})^2}{2} \cdot \tfrac{\deg}{|\mathbb{F}|-N}$$

DEEP Soundness

$$\Pr\left[\Delta\left(\left[\tfrac{\boldsymbol{p}[i]-y}{D[i]-z}\right]_i, \mathsf{RS}\right) < \delta \,\middle|\, \forall f(X) \in \mathsf{list}(\boldsymbol{p})\,,\ f(z) \neq y\right] \leqslant \frac{(\#\mathsf{list})^2}{2} \cdot \frac{\deg}{|\mathbb{F}|-N}$$

$\delta \in$	probability	confidence
$\left[0; \frac{1-\rho}{2}\right)$	negligible	provable security ($+$ simple analysis)
$\left[\frac{1-\rho}{2}; 1-\sqrt{\rho}\right)$	negligible	provable security
$\left[1-\sqrt{\rho}; 1-\rho\right)$	negligible?	conjectural
$[1-\rho; 1]$	???	insecure

Polynomial IOP to Low-Degree IOP (3)

Polynomial IOP:	**Low-Degree IOP:**
polynomial oracles $p(X) \in \mathbb{F}[X]$	codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$
evaluation queries	pointwise maps
	low degree tests

simulate

Prover	Verifier	Prover		Verifier
$p(X)$	$p(X)$	$\boldsymbol{p}$		$\boldsymbol{p}$
	$z \xleftarrow{\$} \mathbb{F}$			$z \xleftarrow{\$} \mathbb{F}$
	$y \leftarrow p(z)$		$\xleftarrow{\quad z \quad}$	
			$\xrightarrow{\quad p(z) \quad}$	
				$\boldsymbol{q} \leftarrow \text{DEEP}(\boldsymbol{p}, p(z), z)$
				$= \left[\frac{\boldsymbol{p}[i]-p(z)}{D[i]-z}\right]_i$

Polynomial IOP to Low-Degree IOP (3)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover	Verifier
$p(X)$	$p(X)$
	$z \xleftarrow{\$} \mathbb{F}$
	$y \leftarrow p(z)$

Prover		Verifier
$\boldsymbol{p}$		$\boldsymbol{p}$
		$z \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad z \quad}$	
	$\xrightarrow{\quad p(z) \quad}$	
		$\boldsymbol{q} \leftarrow \mathsf{DEEP}(\boldsymbol{p}, p(z), z)$
		$= \left[\frac{\boldsymbol{p}[i]-p(z)}{D[i]-z}\right]_i$
		$\Delta(\boldsymbol{p}, \mathsf{RS}) \overset{?}{<} \delta_0$
		$\Delta(\boldsymbol{q}, \mathsf{RS}) \overset{?}{<} \delta_0$

Polynomial IOP to Low-Degree IOP (3)

Polynomial IOP:

polynomial oracles $p(X) \in \mathbb{F}[X]$

evaluation queries

Low-Degree IOP:

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover	Verifier
$p(X)$	$p(X)$
	$z \xleftarrow{\$} \mathbb{F}$
	$y \leftarrow p(z)$

Soundness.

$\varepsilon = \epsilon_{\text{DEEP}}$

Prover		Verifier
$\boldsymbol{p}$		$\boldsymbol{p}$
		$z \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad z \quad}$	
	$\xrightarrow{\quad p(z) \quad}$	

$$\boldsymbol{q} \leftarrow \text{DEEP}(\boldsymbol{p}, p(z), z) = \left[\frac{\boldsymbol{p}[i] - p(z)}{D[i] - z}\right]_i$$

$$\Delta(\boldsymbol{p}, \text{RS}) \overset{?}{<} \delta_0$$

$$\Delta(\boldsymbol{q}, \text{RS}) \overset{?}{<} \delta_0$$

DEEP-ALI + DEEP

1. **Low degree extend** the trace
2. **Evaluate** AIR
3. **Divide** out zerofiers
4. **DEEP** in out-of-domain point

Prover		Verifier
trace T, AIR $\mathcal{C}$		AIR $\mathcal{C}$
$\mathbb{t} \xleftarrow{\text{LDE}} T$		
$\mathbb{q} \leftarrow \mathcal{C} \circ \mathbb{t} / \mathbb{Z}$		
	$\xrightarrow{\mathbb{t},\, \mathbb{q}}$	
		$z \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{z}$	
	$\xrightarrow{\boldsymbol{t}(z),\, \boldsymbol{t}(\omega z)}$	
		$\boldsymbol{q}(z) \leftarrow \mathcal{C} \circ \boldsymbol{t}(z) / \boldsymbol{Z}(z)$
		$\mathbb{u} \leftarrow \left[\frac{\mathbb{t}[i] - \boldsymbol{t}(z)}{D[i] - z}\right] \quad \mathbb{v} \leftarrow \left[\frac{\mathbb{t}[i] - \boldsymbol{t}(\omega z)}{D[i] - \omega z}\right] \quad \mathbb{r} \leftarrow \left[\frac{\mathbb{q}[i] - \boldsymbol{q}(z)}{D[i] - z}\right]$
		$\Delta(\mathbb{x}, \text{RS}) \overset{?}{<} \delta_0$ for $\mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$

DEEP-ALI + DEEP

1. **Low degree extend** the trace
2. **Evaluate** AIR
3. **Divide** out zerofiers
4. **DEEP** in out-of-domain point

Soundness.

$\frac{N-1}{\lvert\mathbb{F}\rvert}$	Schwartz-Zippel
$+\epsilon_{\text{DEEP}}(\delta_0)$	list-decoding

Prover		Verifier
trace T, AIR $\mathcal{C}$		AIR $\mathcal{C}$
$\mathbb{t} \xleftarrow{\text{LDE}} T$		
$\mathbb{q} \leftarrow \mathcal{C} \circ \mathbb{t} / \mathbb{Z}$	$\xrightarrow{\mathbb{t},\, \mathbb{q}}$	
		$z \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{z}$	
	$\xrightarrow{\boldsymbol{t}(z),\, \boldsymbol{t}(\omega z)}$	

$\boldsymbol{q}(z) \leftarrow \mathcal{C} \circ \boldsymbol{t}(z) / \boldsymbol{Z}(z)$

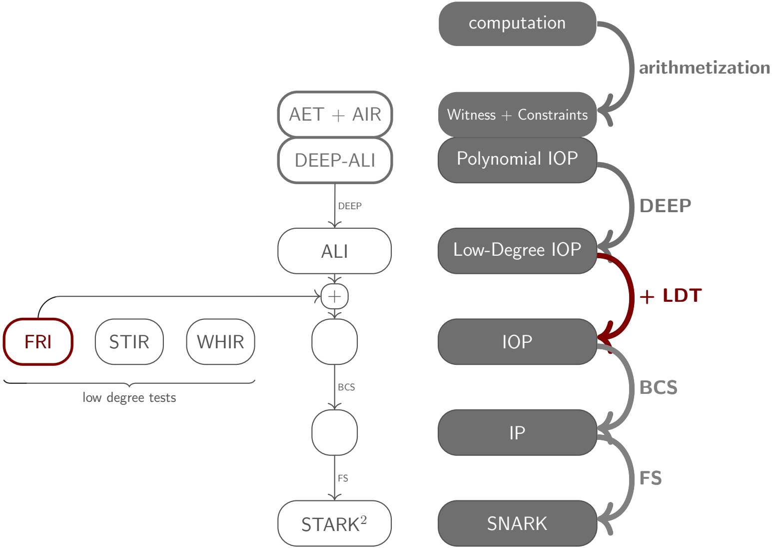
$\mathbb{u} \leftarrow \left[\frac{\mathbb{t}[i] - \boldsymbol{t}(z)}{D[i] - z}\right] \quad \mathbb{v} \leftarrow \left[\frac{\mathbb{t}[i] - \boldsymbol{t}(\omega z)}{D[i] - \omega z}\right] \quad \mathbb{r} \leftarrow \left[\frac{\mathbb{q}[i] - \boldsymbol{q}(z)}{D[i] - z}\right]$

$\Delta(\mathbb{x}, \text{RS}) \overset{?}{<} \delta_0$ for $\mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$

Table of Contents

Table of Contents

STARK Compilation Pipeline (Low-Degree Testing)



Intermezzo: Batching

$$\Delta(\mathbb{x}, \mathsf{RS}) \overset{?}{<} \delta_0 \quad \textsf{for} \quad \mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$$

Intermezzo: Batching

$$\Delta(\mathbb{x}, \mathsf{RS}) \overset{?}{<} \delta_0 \quad \text{for} \quad \mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$$

- $3\mathsf{w} + 2\#\mathcal{C}$ codewords (a lot)

Intermezzo: Batching

$$\Delta(\mathbb{x}, \mathsf{RS}) \stackrel{?}{<} \delta_0 \quad \text{for} \quad \mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$$

- $3\mathsf{w} + 2\#\mathcal{C}$ codewords (a lot)
- Q: *can we batch them?*

Intermezzo: Batching

$$\Delta(\mathbb{x}, \mathsf{RS}) \stackrel{?}{<} \delta_0 \quad \text{for} \quad \mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$$

- $3\mathsf{w} + 2\#\mathcal{C}$ codewords (a lot)
- Q: *can we batch them?* A: yes!

Intermezzo: Batching

$$\Delta(\mathbb{x}, \mathsf{RS}) \stackrel{?}{<} \delta_0 \quad \text{for} \quad \mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$$

- $3\mathsf{w} + 2\#\mathcal{C}$ codewords (a lot)
- Q: *can we batch them?* A: yes!
- Strategy: *random linear combination* ✓

Intermezzo: Batching

$$\Delta(\mathbb{x}, \mathsf{RS}) \overset{?}{<} \delta_0 \quad \text{for} \quad \mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$$

- $3\mathsf{w} + 2\#\mathcal{C}$ codewords (a lot)
- Q: *can we batch them?* A: yes!
- Strategy: *random linear combination* ✓
- Soundness?

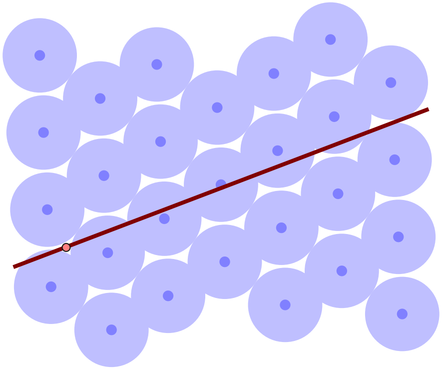
Intermezzo: Batching

$$\Delta(\mathbb{x}, \mathsf{RS}) \overset{?}{<} \delta_0 \quad \text{for} \quad \mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$$

- $3\mathsf{w} + 2\#\mathcal{C}$ codewords (a lot)
- Q: *can we batch them?* A: yes!
- Strategy: *random linear combination* ✓
- Soundness?

$$\Pr_r\left[\Delta((1-r)\cdot \boldsymbol{u} + r\cdot \boldsymbol{v}, \mathsf{RS}) < \delta \,|\, \Delta(\boldsymbol{u}, \mathsf{RS}) > \delta \,\vee\, \Delta(\boldsymbol{v}, \mathsf{RS}) > \delta\right]$$

Reed-Solomon Proximity Gap



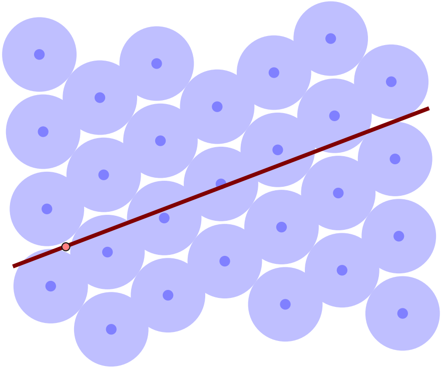
Reed-Solomon Proximity Gap

consider any straight line

let ε be the proportion

of the line δ-close to RS

then $\varepsilon \notin (\epsilon; 1)$



Reed-Solomon Proximity Gap

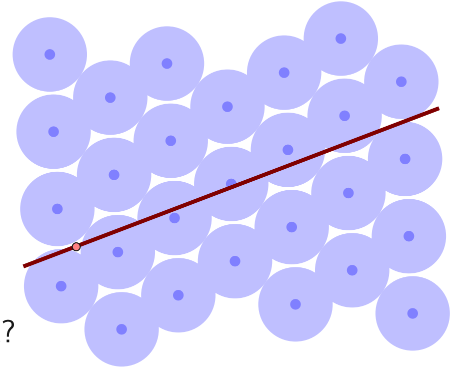
consider any straight line

let ε be the proportion

of the line δ-close to RS

then $\varepsilon \notin (\epsilon; 1)$

for which δ does $\epsilon < 1$ exist?



Reed-Solomon Proximity Gap

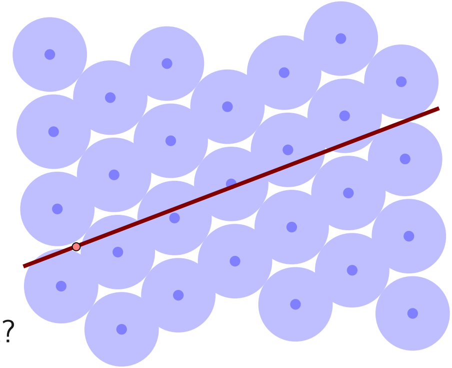
consider any straight line

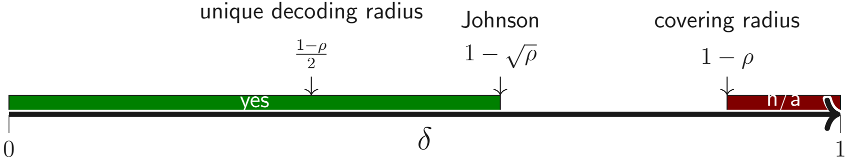
let ε be the proportion

of the line δ-close to RS

then $\varepsilon \notin (\epsilon; 1)$

for which δ does $\epsilon < 1$ exist?





Reed-Solomon Proximity Gap

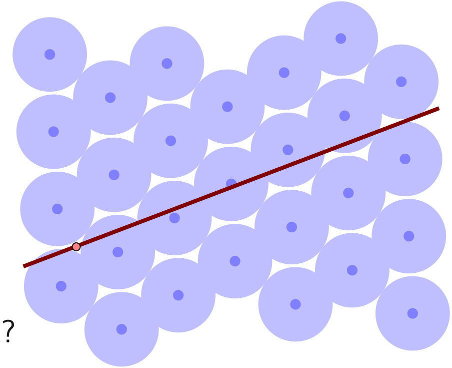
consider any straight line

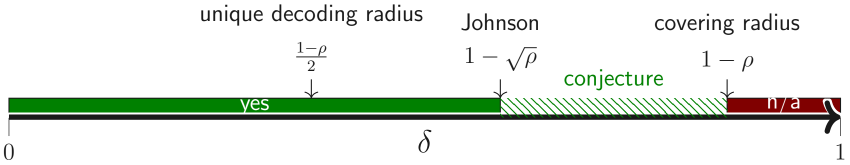
let ε be the proportion

of the line δ-close to RS

then $\varepsilon \notin (\epsilon; 1)$

for which δ does $\epsilon < 1$ exist?





Intermezzo: Batching

$$\Delta(\mathbb{x}, \mathsf{RS}) \overset{?}{<} \delta_0 \quad \text{for} \quad \mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$$

- $3\mathsf{w} + 2\#\mathcal{C}$ codewords (a lot)
- Q: *can we batch them?*

Intermezzo: Batching

$$\Delta(\mathbb{x}, \mathsf{RS}) \overset{?}{<} \delta_0 \quad \text{for} \quad \mathbb{x} \in \{\mathbb{t}, \mathbb{u}, \mathbb{v}, \mathbb{q}, \mathbb{r}\}$$

- $3\mathsf{w} + 2\#\mathcal{C}$ codewords (a lot)
- Q: *can we batch them?* A: yes!
- Strategy: *random linear combination* ✓
- Soundness?

$$\exists(\epsilon_0, \delta_0)\text{-gap}$$

$$\Leftrightarrow$$

$$\Pr_r\left[\Delta((1-r)\cdot \boldsymbol{u} + r \cdot \boldsymbol{v}, \mathsf{RS}) < \delta_0 \,|\, \Delta(\boldsymbol{u}, \mathsf{RS}) > \delta_0 \vee \Delta(\boldsymbol{v}, \mathsf{RS}) > \delta_0\right] \leqslant \epsilon_0$$

</Intermezzo>

Low Degree IOP to Smaller Low Degree IOP

Low Degree IOP

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

for vectors of length N/ρ

Smaller Low Degree IOP

codeword oracles $\boldsymbol{u} \in \mathbb{F}^{N/\rho} \cup \mathbb{F}^{N/2\rho}$

pointwise maps

low degree tests

for vectors of length $N/2\rho$

simulate

Prover Verifier

Prover Verifier

Low Degree IOP to Smaller Low Degree IOP

Low Degree IOP

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

for vectors of length N/ρ

Smaller Low Degree IOP

codeword oracles $\boldsymbol{u} \in \mathbb{F}^{N/\rho} \cup \mathbb{F}^{N/2\rho}$

pointwise maps

low degree tests

for vectors of length $N/2\rho$

simulate

Prover | Verifier

$$\xrightarrow{\boldsymbol{p}}$$

$$\Delta(\boldsymbol{p}, \mathsf{RS}^{N/\rho}) \overset{?}{<} \delta_0$$

Prover | Verifier

Low Degree IOP to Smaller Low Degree IOP

Low Degree IOP

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

for vectors of length N/ρ

Smaller Low Degree IOP

codeword oracles $\boldsymbol{u} \in \mathbb{F}^{N/\rho} \cup \mathbb{F}^{N/2\rho}$

pointwise maps

low degree tests

for vectors of length $N/2\rho$

simulate

Prover Verifier

$$\text{Prover} \xrightarrow{\boldsymbol{p}} \text{Verifier}$$

$$\Delta(\boldsymbol{p}, \mathsf{RS}^{N/\rho}) \overset{?}{<} \delta_0$$

Prover Verifier

$$\text{Prover} \xrightarrow{\boldsymbol{p}} \text{Verifier}$$

$$\boldsymbol{u}, \boldsymbol{v} \leftarrow \mathsf{split}(\boldsymbol{p})$$

$$r \overset{\$}{\leftarrow} \mathbb{F}$$

$$\boldsymbol{c} \leftarrow (1 - r) \cdot \boldsymbol{u} + r \cdot \boldsymbol{v}$$

$$\Delta(\boldsymbol{c}, \mathsf{RS}^{N/2\rho}) \overset{?}{<} \delta_0$$

Low Degree IOP to Smaller Low Degree IOP

Low Degree IOP

codeword oracles $\boldsymbol{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

for vectors of length N/ρ

Smaller Low Degree IOP

codeword oracles $\boldsymbol{u} \in \mathbb{F}^{N/\rho} \cup \mathbb{F}^{N/2\rho}$

pointwise maps

low degree tests

for vectors of length $N/2\rho$

simulate

Prover — Verifier

$\xrightarrow{\boldsymbol{p}}$

$\Delta(\boldsymbol{p}, \mathsf{RS}^{N/\rho}) \overset{?}{<} \delta_0$

Prover — Verifier

$\xrightarrow{\boldsymbol{p}}$

$\boldsymbol{u}, \boldsymbol{v} \leftarrow \boxed{\mathsf{split}(\boldsymbol{p})}$

$r \xleftarrow{\$} \mathbb{F}$

$\boldsymbol{c} \leftarrow (1-r) \cdot \boldsymbol{u} + r \cdot \boldsymbol{v}$

$\Delta(\boldsymbol{c}, \mathsf{RS}^{N/2\rho}) \overset{?}{<} \delta_0$

Split

$\rightarrow$ half the length; same rate

$$\text{split} \;:\; \mathsf{RS}^{N/\rho} \to \mathsf{RS}^{N/2\rho} \times \mathsf{RS}^{N/2\rho}$$

Split

$\rightarrow$ half the length; same rate

$$\text{split} \;:\; \mathsf{RS}^{N/\rho} \rightarrow \mathsf{RS}^{N/2\rho} \times \mathsf{RS}^{N/2\rho}$$
$$f(D) \mapsto (\, f_E(D^2), f_O(D^2) \,)$$

where $f_E(X^2) = \frac{f(X)+f(-X)}{2}$ and $f_O(X^2) = \frac{f(X)-f(-X)}{2X}$

Split

$\rightarrow$ half the length; same rate

$$\begin{aligned}
\mathsf{split}\ :\ & \mathsf{RS}^{N/\rho} \rightarrow \mathsf{RS}^{N/2\rho} \times \mathsf{RS}^{N/2\rho} \\
& f(D) \mapsto (\ f_E(D^2), f_O(D^2)\ ) \\
& \boldsymbol{f} \mapsto \begin{pmatrix} \left[\frac{\boldsymbol{f}[i]+\boldsymbol{f}[i+N/2\rho]}{2}\right]_{i=0}^{N/2\rho} \\ \\ \left[\frac{\boldsymbol{f}[i]-\boldsymbol{f}[i+N/2\rho]}{2D[i]}\right]_{i=0}^{N/2\rho} \end{pmatrix}
\end{aligned}$$

where $f_E(X^2) = \frac{f(X)+f(-X)}{2}$ and $f_O(X^2) = \frac{f(X)-f(-X)}{2X}$

Split

$\to$ half the length; same rate

$$
\begin{aligned}
\text{split} \;:\; & \mathsf{RS}^{N/\rho} \to \mathsf{RS}^{N/2\rho} \times \mathsf{RS}^{N/2\rho} \\
& f(D) \mapsto (\, f_E(D^2), f_O(D^2) \,)
\end{aligned}
$$

where $f_E(X^2) = \frac{f(X)+f(-X)}{2}$ and $f_O(X^2) = \frac{f(X)-f(-X)}{2X}$

$$
\boldsymbol{f} \mapsto \begin{pmatrix} \left[\frac{\boldsymbol{f}[i]+\boldsymbol{f}[i+N/2\rho]}{2}\right]_{i=0}^{N/2\rho} \\ \\ \left[\frac{\boldsymbol{f}[i]-\boldsymbol{f}[i+N/2\rho]}{2D[i]}\right]_{i=0}^{N/2\rho} \end{pmatrix}
$$

Soundness.

$$
\Pr[\Delta(f_E(D^2), \mathsf{RS}^{N/2\rho}) < \delta \wedge \Delta(f_O(D^2), \mathsf{RS}^{N/2\rho}) < \delta \,|\, \Delta(f(D), \mathsf{RS}^{N/\rho}) > \delta]
$$

Split

$\to$ half the length; same rate

$$\begin{aligned} \text{split} \;:\; & \mathsf{RS}^{N/\rho} \to \mathsf{RS}^{N/2\rho} \times \mathsf{RS}^{N/2\rho} \\ & f(D) \mapsto (\, f_E(D^2), f_O(D^2) \,) \end{aligned}$$

where $f_E(X^2) = \frac{f(X)+f(-X)}{2}$ and $f_O(X^2) = \frac{f(X)-f(-X)}{2X}$

$$\boldsymbol{f} \mapsto \begin{pmatrix} \left[\frac{\boldsymbol{f}[i]+\boldsymbol{f}[i+N/2\rho]}{2} \right]_{i=0}^{N/2\rho} \\ \\ \left[\frac{\boldsymbol{f}[i]-\boldsymbol{f}[i+N/2\rho]}{2D[i]} \right]_{i=0}^{N/2\rho} \end{pmatrix}$$

Soundness.

$$\Pr[\Delta(f_E(D^2), \mathsf{RS}^{N/2\rho}) < \delta \wedge \Delta(f_O(D^2), \mathsf{RS}^{N/2\rho}) < \delta \mid \Delta(f(D), \mathsf{RS}^{N/\rho}) > \delta] \;\; = 0$$

– no probability variables $\Rightarrow \Pr \in \{0, 1\}$

– local map is linear over $\mathbb{F}^2$ and invertible, so $0 \leftrightarrow 0$

$\mathsf{split}^{\log N}$

$$\overbrace{\mathsf{split} \circ \mathsf{split} \circ \cdots \circ \mathsf{split}}^{\log N \times}$$

$\text{split}^{\log N}$

$$\overbrace{\text{split} \circ \text{split} \circ \cdots \circ \text{split}}^{\log N \times}$$

Problem: verifier work explodes exponentially

$\mathsf{split}^{\log N}$

$$\overbrace{\mathsf{split} \circ \mathsf{split} \circ \cdots \circ \mathsf{split}}^{\log N \times}$$

Problem: verifier work explodes exponentially

Solution: prover helps

$\text{split}^{\log N}$

$$\overbrace{\text{split} \circ \text{split} \circ \cdots \circ \text{split}}^{\log N \times}$$

Problem: verifier work explodes exponentially

Solution: prover helps

Problem: malicious help

Solution: verifier checks help

FRI: Fast Reed-Solomon IOPP

Prover		Verifier
$\boldsymbol{a}$		$\boldsymbol{a}$
		$r \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad r \quad}$	
$\boldsymbol{a}_E, \boldsymbol{a}_O \leftarrow \mathsf{split}(\boldsymbol{a})$		
$\boldsymbol{b} \leftarrow (1-r) \cdot \boldsymbol{a}_E + r \cdot \boldsymbol{a}_O$		
	$\xrightarrow{\quad \boldsymbol{b} \quad}$	

FRI: Fast Reed-Solomon IOPP

$$
\begin{array}{lcr}
\textsf{Prover} & & \textsf{Verifier} \\
\boldsymbol{a} & & \boldsymbol{a} \\
 & & r \xleftarrow{\$} \mathbb{F} \\
 & \xleftarrow{\quad r \quad} & \\
\boldsymbol{a}_E, \boldsymbol{a}_O \leftarrow \mathsf{split}(\boldsymbol{a}) & & \\
\boldsymbol{b} \leftarrow (1-r)\cdot \boldsymbol{a}_E + r \cdot \boldsymbol{a}_O & & \\
 & \xrightarrow{\quad \boldsymbol{b} \quad} & \\
 & & s \xleftarrow{\$} \mathbb{F} \\
 & \xleftarrow{\quad s \quad} & \\
\boldsymbol{b}_E, \boldsymbol{b}_O \leftarrow \mathsf{split}(\boldsymbol{b}) & & \\
\boldsymbol{c} \leftarrow (1-s)\cdot \boldsymbol{b}_E + s \cdot \boldsymbol{b}_O & & \\
 & \xrightarrow{\quad \boldsymbol{c} \quad} & \\
\vdots & \vdots & \vdots \\
 & \xrightarrow{\quad \boldsymbol{d} \quad} & \\
 & & \boldsymbol{d} \stackrel{?}{\in} \mathsf{RS}^{1/\rho}
\end{array}
$$

FRI: Fast Reed-Solomon IOPP

commit phase

Prover		Verifier
$\boldsymbol{a}$		$\boldsymbol{a}$
		$r \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad r \quad}$	
$\boldsymbol{a}_E, \boldsymbol{a}_O \leftarrow \mathsf{split}(\boldsymbol{a})$		
$\boldsymbol{b} \leftarrow (1-r) \cdot \boldsymbol{a}_E + r \cdot \boldsymbol{a}_O$		
	$\xrightarrow{\quad \boldsymbol{b} \quad}$	
		$s \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad s \quad}$	
$\boldsymbol{b}_E, \boldsymbol{b}_O \leftarrow \mathsf{split}(\boldsymbol{b})$		
$\boldsymbol{c} \leftarrow (1-s) \cdot \boldsymbol{b}_E + s \cdot \boldsymbol{b}_O$		
	$\xrightarrow{\quad \boldsymbol{c} \quad}$	
$\vdots$	$\vdots$	$\vdots$
	$\xrightarrow{\quad \boldsymbol{d} \quad}$	
		$\boldsymbol{d} \overset{?}{\in} \mathsf{RS}^{1/\rho}$

FRI: Fast Reed-Solomon IOPP

Prover		Verifier
$\boldsymbol{a}$		$\boldsymbol{a}$
		$r \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad r \quad}$	
$\boldsymbol{a}_E, \boldsymbol{a}_O \leftarrow \mathsf{split}(\boldsymbol{a})$		
$\boldsymbol{b} \leftarrow (1-r)\cdot \boldsymbol{a}_E + r \cdot \boldsymbol{a}_O$		
	$\xrightarrow{\quad \boldsymbol{b} \quad}$	
		$s \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad s \quad}$	
$\boldsymbol{b}_E, \boldsymbol{b}_O \leftarrow \mathsf{split}(\boldsymbol{b})$		
$\boldsymbol{c} \leftarrow (1-s)\cdot \boldsymbol{b}_E + s \cdot \boldsymbol{b}_O$		
	$\xrightarrow{\quad \boldsymbol{c} \quad}$	
$\vdots$	$\vdots$	$\vdots$
	$\xrightarrow{\quad \boldsymbol{d} \quad}$	
		$\boldsymbol{d} \overset{?}{\in} \mathsf{RS}^{1/\rho}$

commit phase

query phase

$i_0, \ldots, i_{t-1} \xleftarrow{\$} \{0, \ldots, N/\rho - 1\}$

for $\boldsymbol{x} \in \{\boldsymbol{b}, \boldsymbol{c}, \ldots, \boldsymbol{d}\}$:

  for $i \in \{i_0, \ldots, i_{t-1}\}$:

    verify split+fold for $\boldsymbol{x}$ in i modulo length

FRI: Fast Reed-Solomon IOPP

Prover		Verifier
$\boldsymbol{a}$		$\boldsymbol{a}$
		$r \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad r \quad}$	
$\boldsymbol{a}_E, \boldsymbol{a}_O \leftarrow \mathsf{split}(\boldsymbol{a})$		
$\boldsymbol{b} \leftarrow (1-r) \cdot \boldsymbol{a}_E + r \cdot \boldsymbol{a}_O$		
	$\xrightarrow{\quad \boldsymbol{b} \quad}$	
		$s \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad s \quad}$	
$\boldsymbol{b}_E, \boldsymbol{b}_O \leftarrow \mathsf{split}(\boldsymbol{b})$		
$\boldsymbol{c} \leftarrow (1-s) \cdot \boldsymbol{b}_E + s \cdot \boldsymbol{b}_O$		
	$\xrightarrow{\quad \boldsymbol{c} \quad}$	
$\vdots$	$\vdots$	$\vdots$
	$\xrightarrow{\quad \boldsymbol{d} \quad}$	

commit phase

query phase

$\boldsymbol{d} \overset{?}{\in} \mathsf{RS}^{1/\rho}$

$i_0, \ldots, i_{t-1} \xleftarrow{\$} \{0, \ldots, N/\rho - 1\}$

for $\boldsymbol{x} \in \{\boldsymbol{b}, \boldsymbol{c}, \ldots, \boldsymbol{d}\}$:

for $i \in \{i_0, \ldots, i_{t-1}\}$:

verify split+fold for $\boldsymbol{x}$ in i modulo length

Soundness.

FRI: Fast Reed-Solomon IOPP

Prover		Verifier
$\boldsymbol{a}$		$\boldsymbol{a}$
		$r \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad r \quad}$	
$\boldsymbol{a}_E, \boldsymbol{a}_O \leftarrow \mathsf{split}(\boldsymbol{a})$		
$\boldsymbol{b} \leftarrow (1-r)\cdot \boldsymbol{a}_E + r \cdot \boldsymbol{a}_O$		
	$\xrightarrow{\quad \boldsymbol{b} \quad}$	
		$s \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{\quad s \quad}$	
$\boldsymbol{b}_E, \boldsymbol{b}_O \leftarrow \mathsf{split}(\boldsymbol{b})$		
$\boldsymbol{c} \leftarrow (1-s)\cdot \boldsymbol{b}_E + s \cdot \boldsymbol{b}_O$		
	$\xrightarrow{\quad \boldsymbol{c} \quad}$	
$\vdots$	$\vdots$	$\vdots$
	$\xrightarrow{\quad \boldsymbol{d} \quad}$	

commit phase

query phase

$\boldsymbol{d} \overset{?}{\in} \mathsf{RS}^{1/\rho}$

$i_0, \ldots, i_{t-1} \xleftarrow{\$} \{0, \ldots, N/\rho - 1\}$

for $\boldsymbol{x} \in \{\boldsymbol{b}, \boldsymbol{c}, \ldots, \boldsymbol{d}\}$:

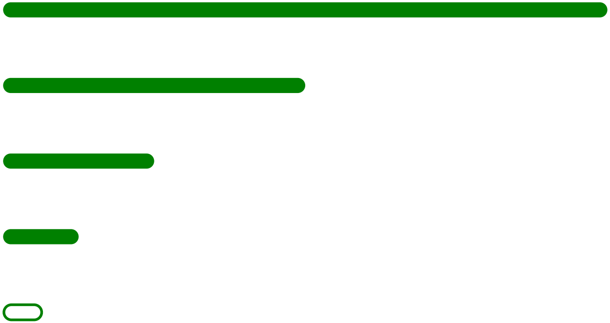
for $i \in \{i_0, \ldots, i_{t-1}\}$:

verify split+fold for $\boldsymbol{x}$ in i modulo length

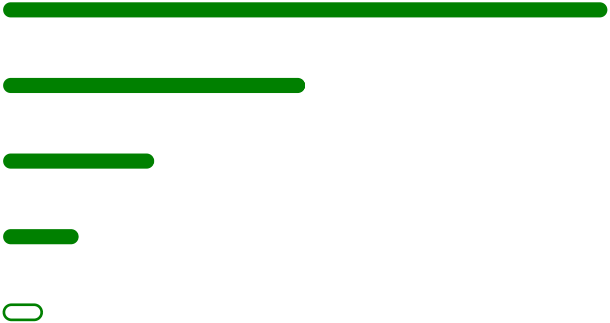
Soundness.

- r rounds
- r $\cdot\, \epsilon_0$ (bad folding — proximity gap)
- $(1-\delta_0)^t$ (bad indices)

FRI: Visual Intuition



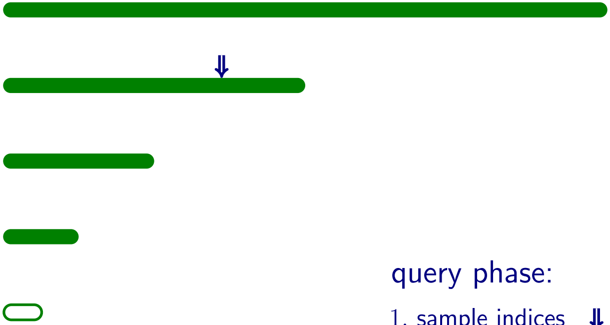
FRI: Visual Intuition



commit phase:

1. commit to all codewords
2. test degree of last codeword

FRI: Visual Intuition



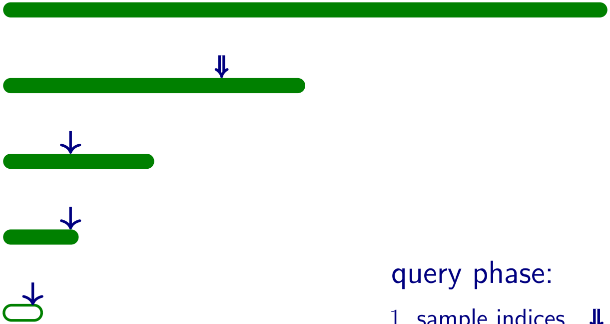
query phase:

1. sample indices $\Downarrow$

commit phase:

1. commit to all codewords
2. test degree of last codeword

FRI: Visual Intuition



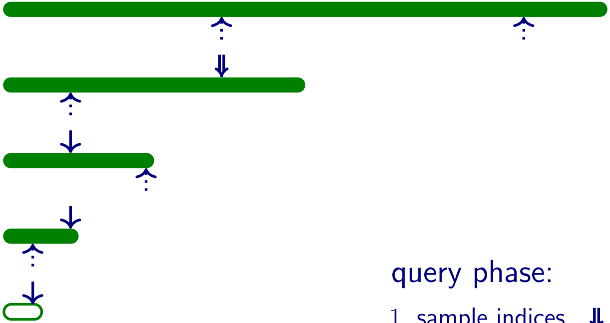
query phase:

1. sample indices $\Downarrow$
2. indices cascade modulo length $\downarrow$

commit phase:

1. commit to all codewords
2. test degree of last codeword

FRI: Visual Intuition



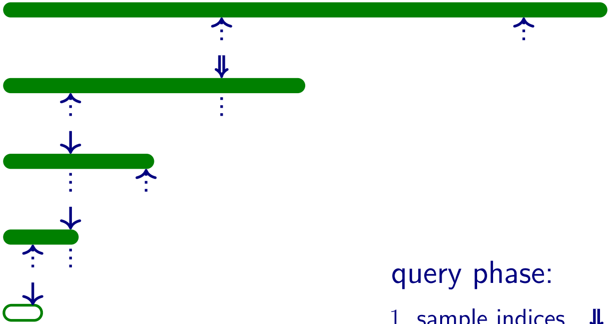
query phase:

1. sample indices ⇓
2. indices cascade modulo length ↓
3. query codewords

commit phase:

1. commit to all codewords
2. test degree of last codeword

FRI: Visual Intuition



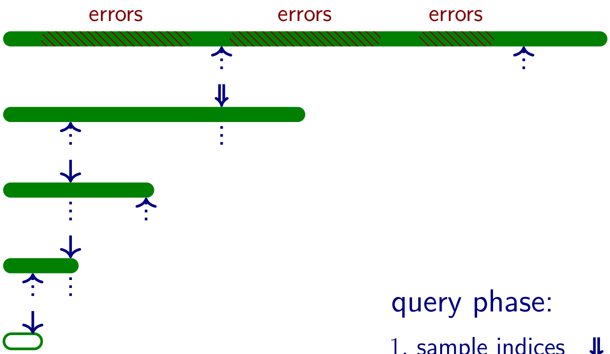
commit phase:

1. commit to all codewords
2. test degree of last codeword

query phase:

1. sample indices $\Downarrow$
2. indices cascade modulo length $\downarrow$
3. query codewords
3. check split+fold $\Downarrow$ $\downarrow$
4. reuse result $\vdots$

FRI: Visual Intuition



Soundness:

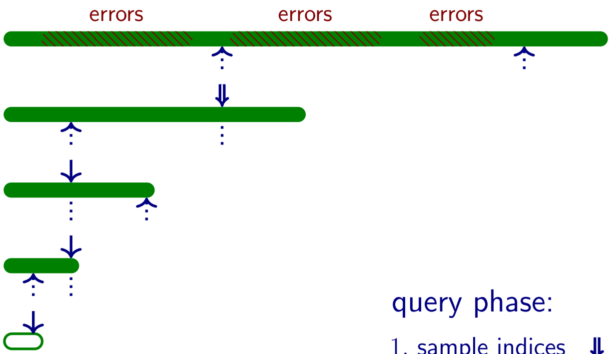
query phase:

1. sample indices ⇓
2. indices cascade modulo length ↓
3. query codewords
3. check split+fold ⇓ ↓
4. reuse result ⋮

commit phase:

1. commit to all codewords
2. test degree of last codeword

FRI: Visual Intuition



Soundness:

probability that:

– 1 index points to error: $\geqslant \delta$

– 1 index points to non-error: $\leqslant 1 - \delta$

– all t indices point to non-errors: $\leqslant (1-\delta)^t$

query phase:

1. sample indices $\Downarrow$
2. indices cascade modulo length $\downarrow$
3. query codewords
3. check split+fold $\Downarrow$ $\downarrow$
4. reuse result $\vdots$

commit phase:

1. commit to all codewords
2. test degree of last codeword

DEEP-ALI + DEEP + FRI

Prover		Verifier
trace T, AIR $\mathcal{C}$		AIR $\mathcal{C}$
$\mathfrak{t} \xleftarrow{\text{LDE}} T$		
$\mathfrak{q} \leftarrow \mathcal{C} \circ \mathfrak{t} / \mathbb{Z}$	$\xrightarrow{\mathfrak{t},\, \mathfrak{q}}$	
		$z \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{z}$	
	$\xrightarrow{\boldsymbol{t}(z),\, \boldsymbol{t}(\omega z)}$	
$\boldsymbol{q}(z) \leftarrow \mathcal{C} \circ \boldsymbol{t}(z) / \boldsymbol{Z}(z)$		$\boldsymbol{q}(z) \leftarrow \mathcal{C} \circ \boldsymbol{t}(z) / \boldsymbol{Z}(z)$
$\mathfrak{u} \leftarrow \left[\frac{\mathfrak{t}[i]-\boldsymbol{t}(z)}{D[i]-z}\right]$ $\mathfrak{v} \leftarrow \left[\frac{\mathfrak{t}[i]-\boldsymbol{t}(\omega z)}{D[i]-\omega z}\right]$ $\mathfrak{r} \leftarrow \left[\frac{\mathfrak{q}[i]-\boldsymbol{q}(z)}{D[i]-z}\right]$		$\mathfrak{u} \leftarrow \left[\frac{\mathfrak{t}[i]-\boldsymbol{t}(z)}{D[i]-z}\right]$ $\mathfrak{v} \leftarrow \left[\frac{\mathfrak{t}[i]-\boldsymbol{t}(\omega z)}{D[i]-\omega z}\right]$ $\mathfrak{r} \leftarrow \left[\frac{\mathfrak{q}[i]-\boldsymbol{q}(z)}{D[i]-z}\right]$
		$\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{3 \cdot \mathsf{w} + 2 \cdot \# \mathcal{C}}$
	$\xleftarrow{\boldsymbol{r}}$	
$\boldsymbol{c} \leftarrow \boldsymbol{r}^{\mathsf{T}}(\mathfrak{t} \vert \mathfrak{q} \vert \mathfrak{u} \vert \mathfrak{v} \vert \mathfrak{r})$		
	$\xrightarrow{\boldsymbol{c}}$	
		$s \xleftarrow{\$} \mathbb{F}$
	$\xleftarrow{s}$	
$\boldsymbol{c}_E, \boldsymbol{c}_O \leftarrow \text{split}(\boldsymbol{c})$		
$\boldsymbol{c} \leftarrow (1-s) \cdot \boldsymbol{c}_E + s \cdot \boldsymbol{c}_O$		
	$\xrightarrow{\boldsymbol{c}}$	
		last $\boldsymbol{c} \overset{?}{\in}$ RS
		$i_0, \ldots, i_{t-1} \xleftarrow{\$} \{0, \ldots, N/\rho - 1\}$
		for all indices i, all codewords $\boldsymbol{c}$ except first:
		verify split+fold for $\boldsymbol{c}$ in i

DEEP-ALI + DEEP + FRI

Prover		Verifier	Soundness.	
trace T, AIR $\mathcal{C}$		AIR $\mathcal{C}$	$\frac{N-1}{\lvert\mathbb{F}\rvert}$	Schwartz-Zippel
$\mathbb{t} \xleftarrow{\text{LDE}} T$			$+\epsilon_{\text{DEEP}}(\delta_0)$	list-decoding
$\mathbb{q} \leftarrow \mathcal{C} \circ \mathbb{t} / \mathbb{Z}$	$\xrightarrow{\mathbb{t},\, \mathbb{q}}$			
		$z \xleftarrow{\$} \mathbb{F}$		
	$\xleftarrow{z}$			
	$\xrightarrow{\boldsymbol{t}(z),\, \boldsymbol{t}(\omega z)}$			
$\boldsymbol{q}(z) \leftarrow \mathcal{C} \circ \boldsymbol{t}(z) / \boldsymbol{Z}(z)$		$\boldsymbol{q}(z) \leftarrow \mathcal{C} \circ \boldsymbol{t}(z) / \boldsymbol{Z}(z)$		
$\mathbb{u} \leftarrow \left[\frac{\mathbb{t}[i]-\boldsymbol{t}(z)}{D[i]-z}\right]$ $\mathbb{v} \leftarrow \left[\frac{\mathbb{t}[i]-\boldsymbol{t}(\omega z)}{D[i]-\omega z}\right]$ $\mathbb{r} \leftarrow \left[\frac{\mathbb{q}[i]-\boldsymbol{q}(z)}{D[i]-z}\right]$		$\mathbb{u} \leftarrow \left[\frac{\mathbb{t}[i]-\boldsymbol{t}(z)}{D[i]-z}\right]$ $\mathbb{v} \leftarrow \left[\frac{\mathbb{t}[i]-\boldsymbol{t}(\omega z)}{D[i]-\omega z}\right]$ $\mathbb{r} \leftarrow \left[\frac{\mathbb{q}[i]-\boldsymbol{q}(z)}{D[i]-z}\right]$		
		$\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{3 \cdot \mathrm{w} + 2 \cdot \#\mathcal{C}}$		
	$\xleftarrow{\boldsymbol{r}}$			
$\boldsymbol{c} \leftarrow \boldsymbol{r}^{\mathsf{T}}(\mathbb{t} \vert \mathbb{q} \vert \mathbb{u} \vert \mathbb{v} \vert \mathbb{r})$				
	$\xrightarrow{\boldsymbol{c}}$			
		$s \xleftarrow{\$} \mathbb{F}$		
	$\xleftarrow{s}$			
$\boldsymbol{c}_E, \boldsymbol{c}_O \leftarrow \mathsf{split}(\boldsymbol{c})$				
$\boldsymbol{c} \leftarrow (1-s) \cdot \boldsymbol{c}_E + s \cdot \boldsymbol{c}_O$				
	$\xrightarrow{\boldsymbol{c}}$			
		last $\boldsymbol{c} \overset{?}{\in}$ RS		
		$i_0, \ldots, i_{t-1} \xleftarrow{\$} \{0, \ldots, N/\rho - 1\}$		
		for all indices i, all codewords $\boldsymbol{c}$ except first:		
		verify split+fold for $\boldsymbol{c}$ in i		

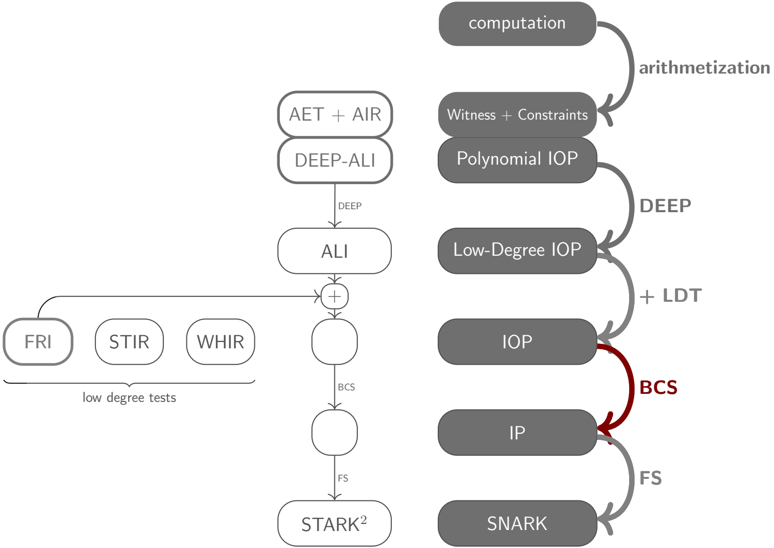
DEEP-ALI + DEEP + FRI

Prover		Verifier	Soundness.	
trace T, AIR $\mathcal{C}$		AIR $\mathcal{C}$	$\frac{N-1}{\lvert\mathbb{F}\rvert}$	Schwartz-Zippel
$\mathbb{t} \xleftarrow{\text{LDE}} T$			$+\epsilon_{\text{DEEP}}(\delta_0)$	list-decoding
$\mathbb{q} \leftarrow \mathcal{C} \circ \mathbb{t}/\mathbb{Z}$			$+(3 \cdot \mathsf{w} + 2 \cdot \#\mathcal{C}) \cdot \epsilon_{\text{GAP}}(\delta_0)$	proximity gap for batch
	$\xrightarrow{\quad \mathbb{t}, \mathbb{q} \quad}$		$+\mathsf{r} \cdot \epsilon_{\text{GAP}}(\delta_0)$	proximity gap for folding
		$z \xleftarrow{\$} \mathbb{F}$	$+(1-\delta_0)^t$	bad indices
	$\xleftarrow{\quad z \quad}$			
	$\xrightarrow{\quad \boldsymbol{t}(z), \boldsymbol{t}(\omega z) \quad}$			
$\boldsymbol{q}(z) \leftarrow \mathcal{C} \circ \boldsymbol{t}(z)/\boldsymbol{Z}(z)$		$\boldsymbol{q}(z) \leftarrow \mathcal{C} \circ \boldsymbol{t}(z)/\boldsymbol{Z}(z)$		
$\mathbb{u} \leftarrow \left[\frac{\mathbb{t}[i]-\boldsymbol{t}(z)}{D[i]-z}\right]$ $\mathbb{v} \leftarrow \left[\frac{\mathbb{t}[i]-\boldsymbol{t}(\omega z)}{D[i]-\omega z}\right]$ $\mathbb{r} \leftarrow \left[\frac{\mathbb{q}[i]-\boldsymbol{q}(z)}{D[i]-z}\right]$		$\mathbb{u} \leftarrow \left[\frac{\mathbb{t}[i]-\boldsymbol{t}(z)}{D[i]-z}\right]$ $\mathbb{v} \leftarrow \left[\frac{\mathbb{t}[i]-\boldsymbol{t}(\omega z)}{D[i]-\omega z}\right]$ $\mathbb{r} \leftarrow \left[\frac{\mathbb{q}[i]-\boldsymbol{q}(z)}{D[i]-z}\right]$		
		$\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^{3\cdot\mathsf{w}+2\cdot\#\mathcal{C}}$		
	$\xleftarrow{\quad \boldsymbol{r} \quad}$			
$\boldsymbol{c} \leftarrow \boldsymbol{r}^{\mathsf{T}}(\mathbb{t}\vert\mathbb{q}\vert\mathbb{u}\vert\mathbb{v}\vert\mathbb{r})$				
	$\xrightarrow{\quad \boldsymbol{c} \quad}$			
		$s \xleftarrow{\$} \mathbb{F}$		
	$\xleftarrow{\quad s \quad}$			
$\boldsymbol{c}_E, \boldsymbol{c}_O \leftarrow \text{split}(\boldsymbol{c})$				
$\boldsymbol{c} \leftarrow (1-s) \cdot \boldsymbol{c}_E + s \cdot \boldsymbol{c}_O$				
	$\xrightarrow{\quad \boldsymbol{c} \quad}$			
		last $\boldsymbol{c} \overset{?}{\in}$ RS		
		$i_0, \ldots, i_{t-1} \xleftarrow{\$} \{0, \ldots, N/\rho - 1\}$		
		for all indices i, all codewords $\boldsymbol{c}$ except first:		
		verify split+fold for $\boldsymbol{c}$ in i		

Table of Contents

Table of Contents

STARK Compilation Pipeline (BCS Transform)



IOP to IP

IOP

codeword oracles $\boldsymbol{c} \in \bigcup_i \in \mathbb{F}^{N/2^i \rho}$

point queries

IP

random oracle H : $\{0,1\}^* \to \{0,1\}^\star$

Q evaluation queries

simulate

$\overbrace{\text{Prover} \qquad \text{Verifier}}$ $\qquad$ $\overbrace{\text{Prover} \qquad \text{Verifier}}$

IOP to IP

IOP

codeword oracles $\boldsymbol{c} \in \bigcup_i \in \mathbb{F}^{N/2^i\rho}$

point queries

IP

random oracle H : $\{0,1\}^* \to \{0,1\}^\star$

Q evaluation queries

simulate

Prover | Verifier

$\boldsymbol{c}$

$\xrightarrow{\boldsymbol{c}}$

$\boldsymbol{c}[i]$

Prover | Verifier

IOP to IP

IOP

codeword oracles $\boldsymbol{c} \in \bigcup_i \in \mathbb{F}^{N/2^i \rho}$

point queries

IP

random oracle H : $\{0,1\}^* \to \{0,1\}^\star$

Q evaluation queries

simulate

$\overbrace{\text{Prover} \qquad\qquad \text{Verifier}}$

$\boldsymbol{c}$

$\xrightarrow{\boldsymbol{c}}$

$\boldsymbol{c}[i]$

$\overbrace{\text{Prover} \qquad\qquad \text{Verifier}}$

$\boldsymbol{c}$

$tree \leftarrow \text{Merkle}(\boldsymbol{c})$

$\xrightarrow{tree.root}$

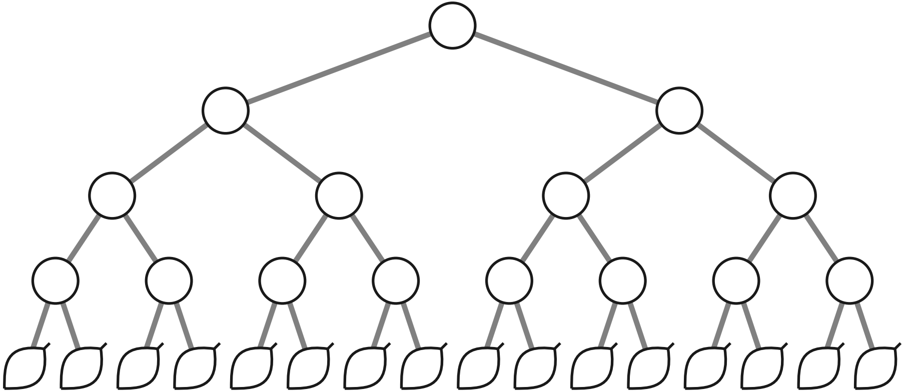
$\xleftarrow{i}$

$\xrightarrow{\boldsymbol{c}[i],\, tree.path(i)}$

$\text{verify}(root, i, path, \boldsymbol{c}[i])$

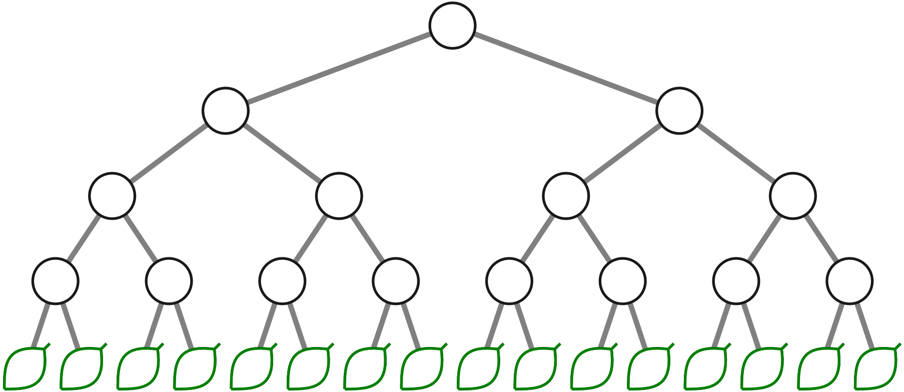
BCS Transform

Merkle Tree



$$parent = \mathsf{H}(left\,child \| right\,child)$$

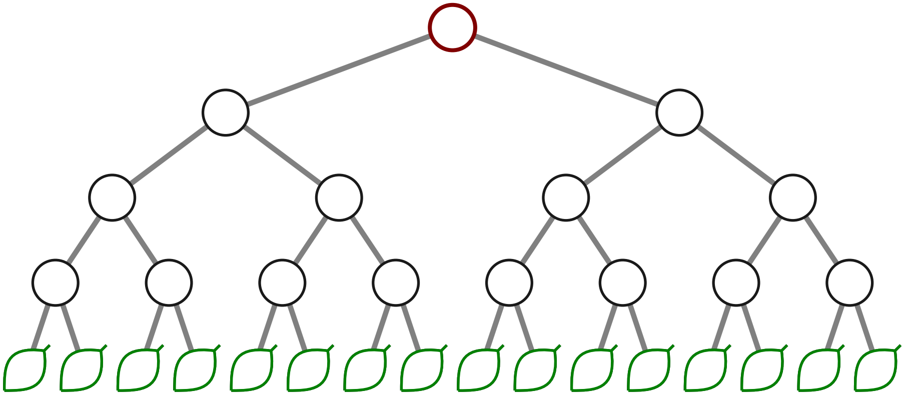
Merkle Tree



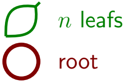
$parent = \mathsf{H}(left\,child \| right\,child)$

n leafs

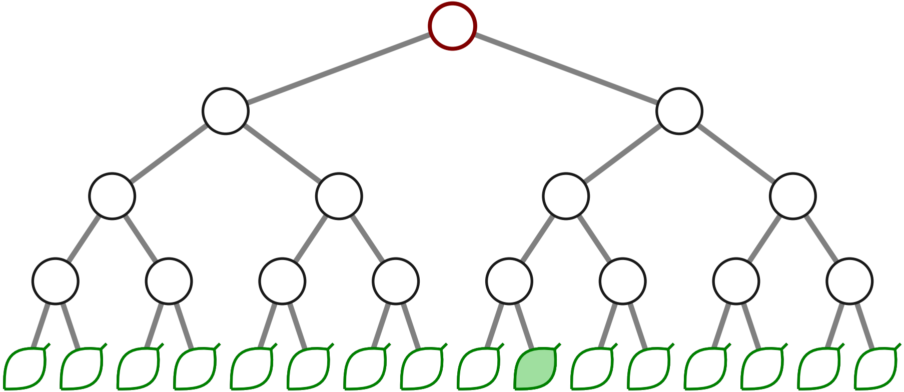
Merkle Tree



$$parent = \mathsf{H}(left\,child \| right\,child)$$

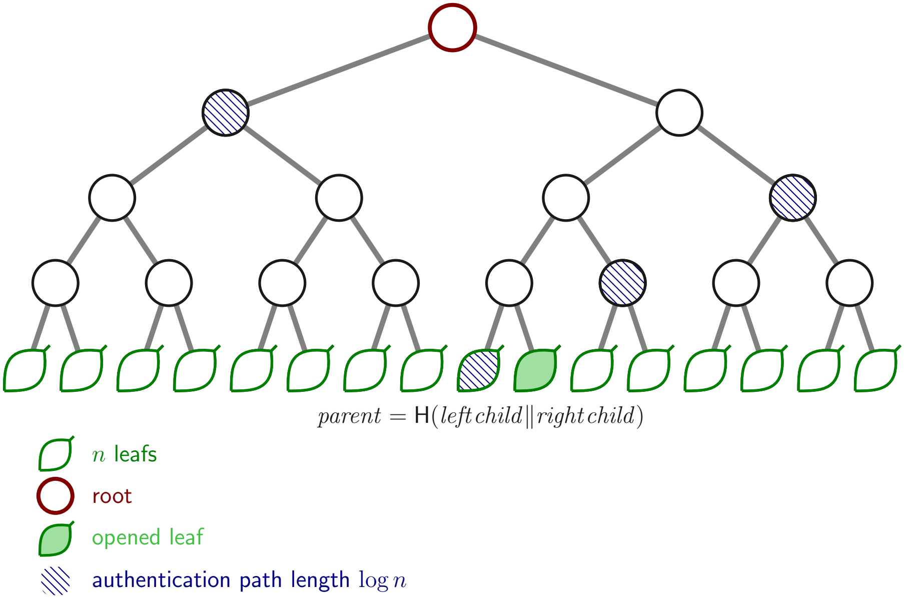


Merkle Tree



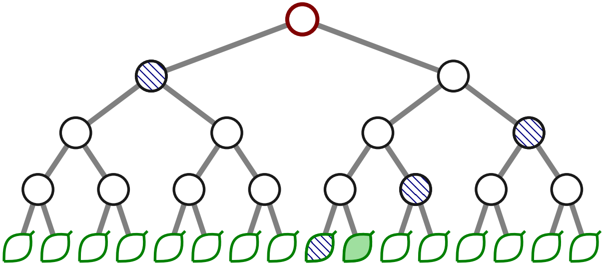
$$parent = \mathsf{H}(left\,child \| right\,child)$$

Merkle Tree

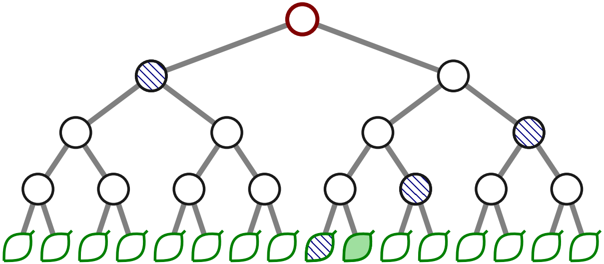


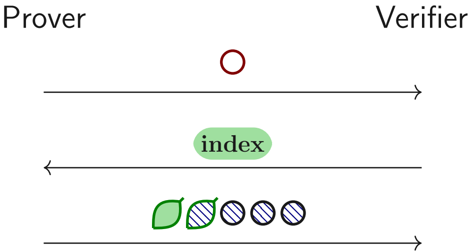
$parent = \mathsf{H}(left\,child \| right\,child)$
n leafs
root
opened leaf
authentication path length $\log n$

Merkle Tree Commitment Scheme

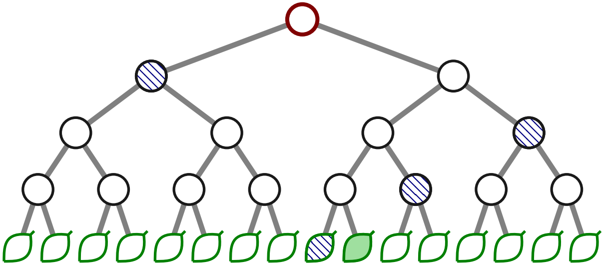


Merkle Tree Commitment Scheme

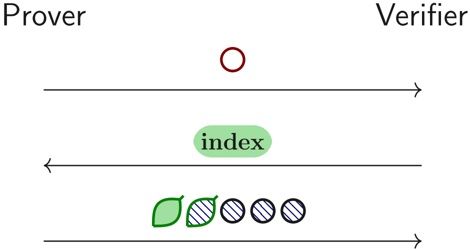




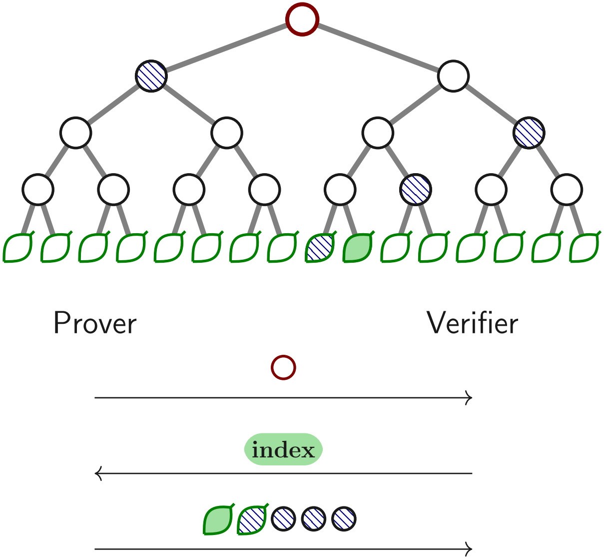
Merkle Tree Commitment Scheme



Soundness.



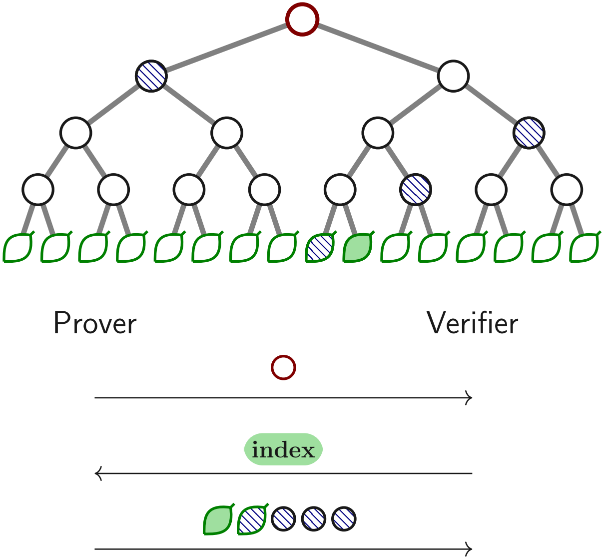
Merkle Tree Commitment Scheme



Soundness.

$\Pr[\text{Verifier}\checkmark \mid \textit{bad leaf}]$

Merkle Tree Commitment Scheme

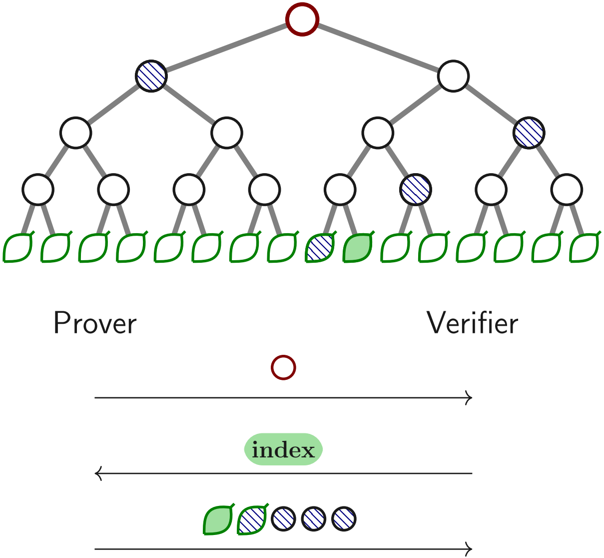


Soundness.

$\Pr[\text{Verifier}\checkmark \mid \textit{bad leaf}]$

$\leqslant \Pr[\textit{hash collision}]$

Merkle Tree Commitment Scheme



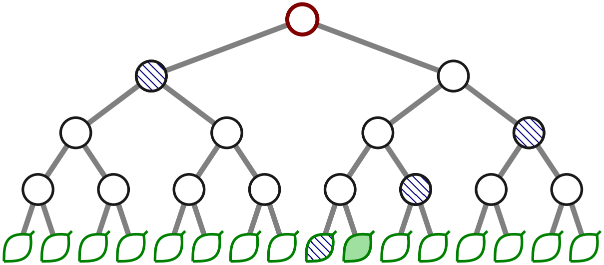
Soundness.

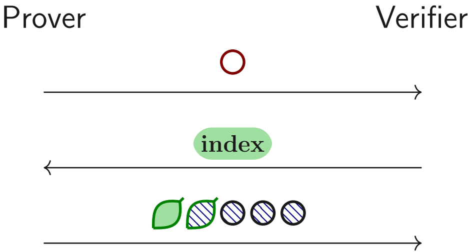
$\Pr[\text{Verifier}\checkmark \mid \textit{bad leaf}]$

$\leqslant \Pr[\textit{hash collision}]$

$= \frac{Q\cdot(Q-1)}{2^\lambda}$

Merkle Tree Commitment Scheme





Soundness.

$\Pr[\text{Verifier}\checkmark \mid \textit{bad leaf}]$

$\leqslant \Pr[\textit{hash collision}]$

$= \frac{Q\cdot(Q-1)}{2^\lambda}$

Q: # hash evaluations

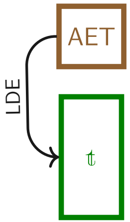
λ: # bits in output

STARK Diagram

$\boxed{\text{AET}}$

algebraic execution trace

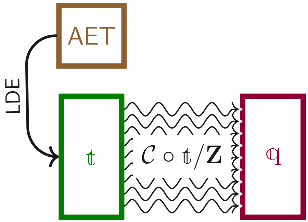
STARK Diagram



low-degree extension

low-degree extended trace

STARK Diagram

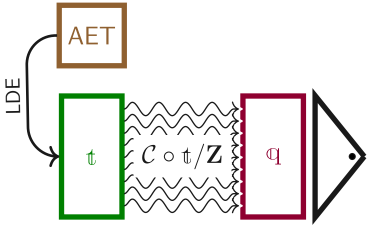


composition with AIR constraints

division by zerofiers

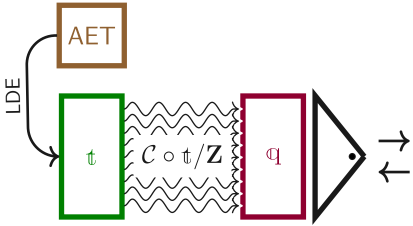
quotients

STARK Diagram



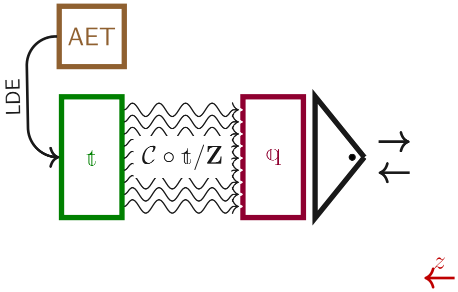
build Merkle tree

STARK Diagram



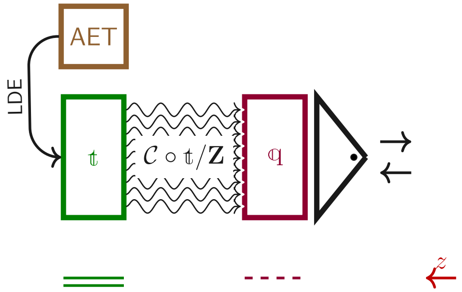
interact with verifier

STARK Diagram



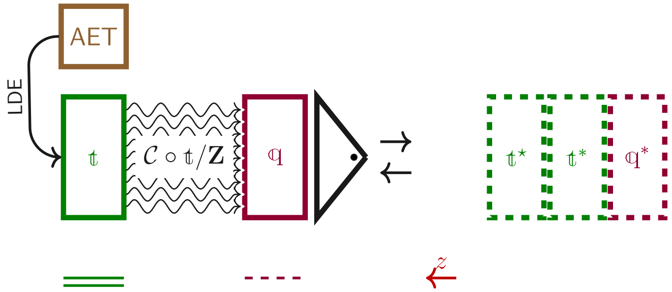
sample out-of-domain point

STARK Diagram



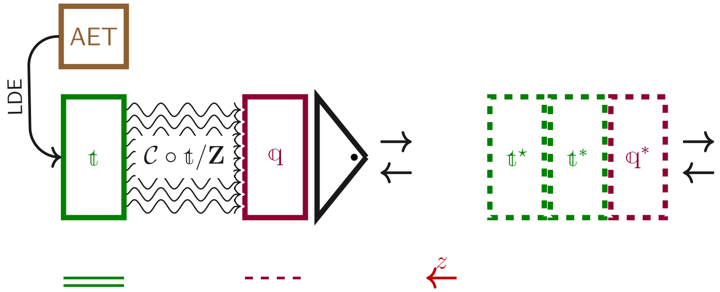
produce out-of-domain rows

STARK Diagram



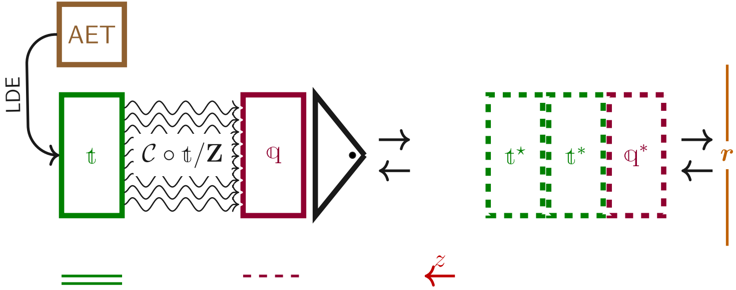
apply DEEP update

STARK Diagram



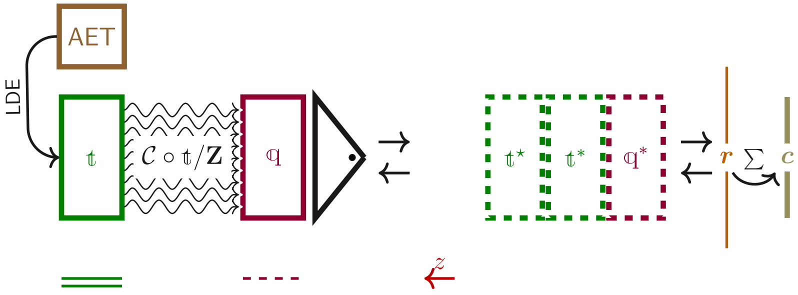
interact with verifier

STARK Diagram



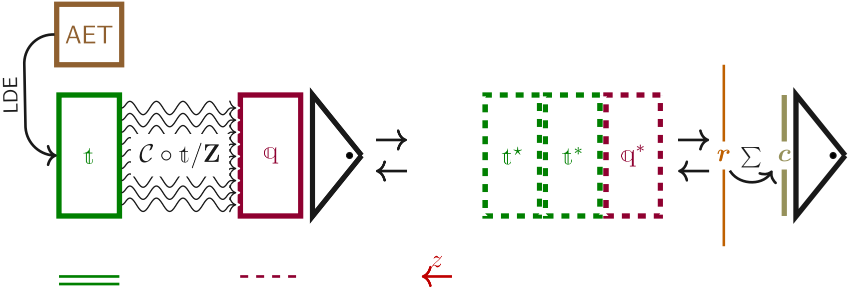
sample weights

STARK Diagram



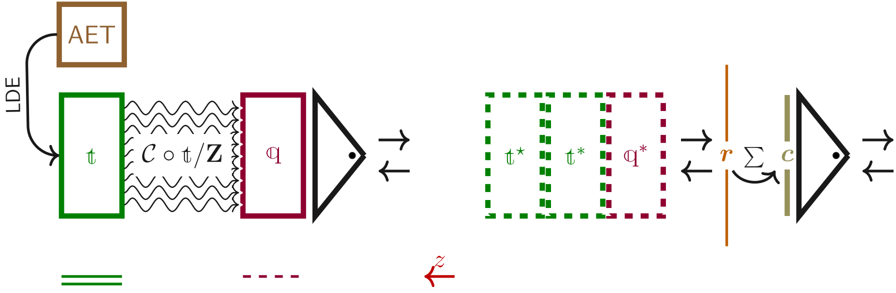
random linear combination

STARK Diagram



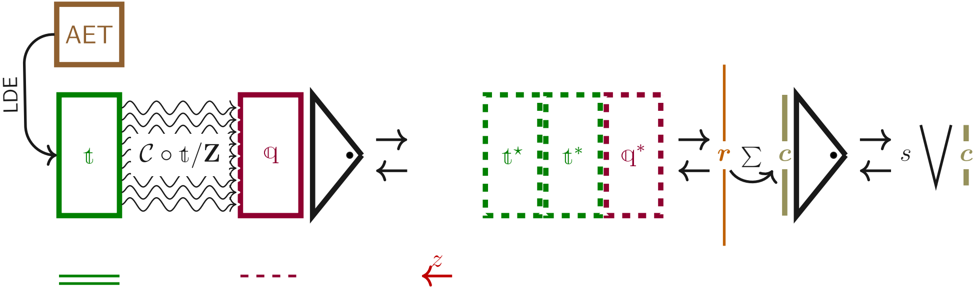
build Merkle tree

STARK Diagram



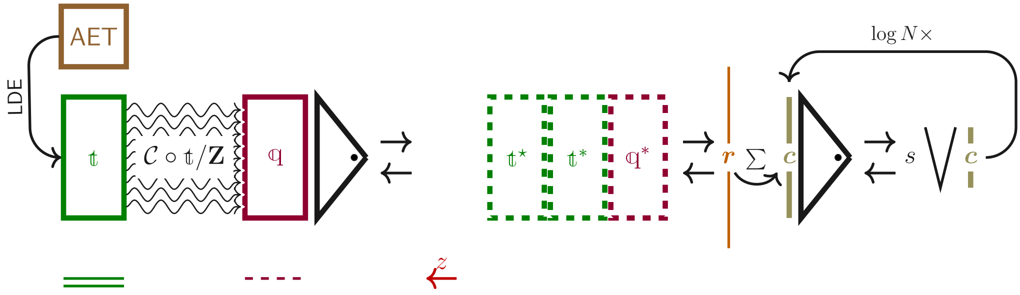
interact with verifier

STARK Diagram



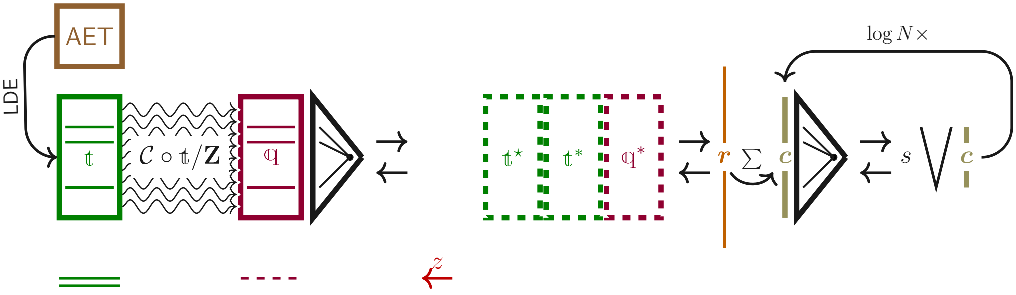
split-and-fold

STARK Diagram



rinse and repeat

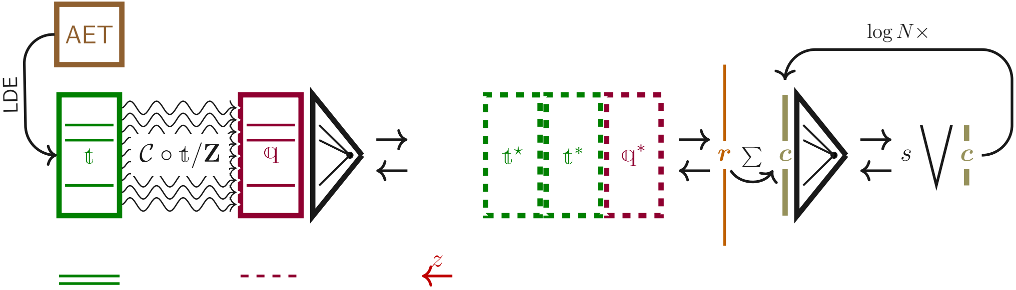
STARK Diagram



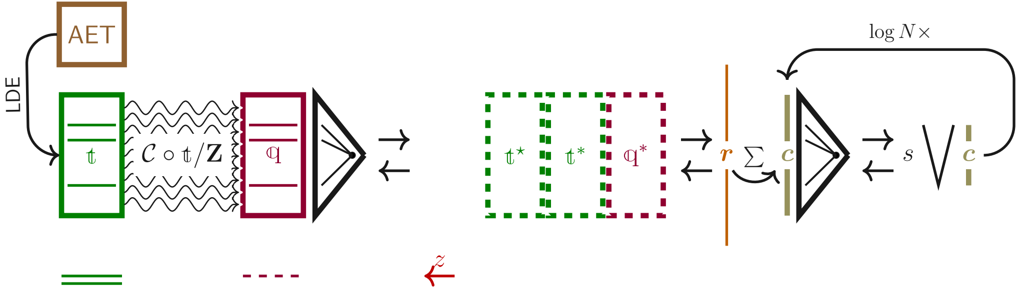
obtain FRI indices

open indicated rows

STARK Diagram

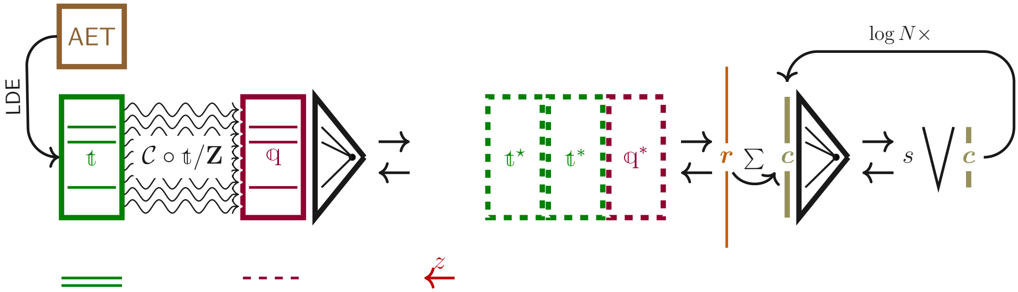


STARK Diagram



Soundness.

STARK Diagram



Soundness.

DEEP-ALI $\{ \quad \frac{N-1}{|\mathbb{F}|} + \epsilon_{\mathsf{DEEP}}(\delta_0)$

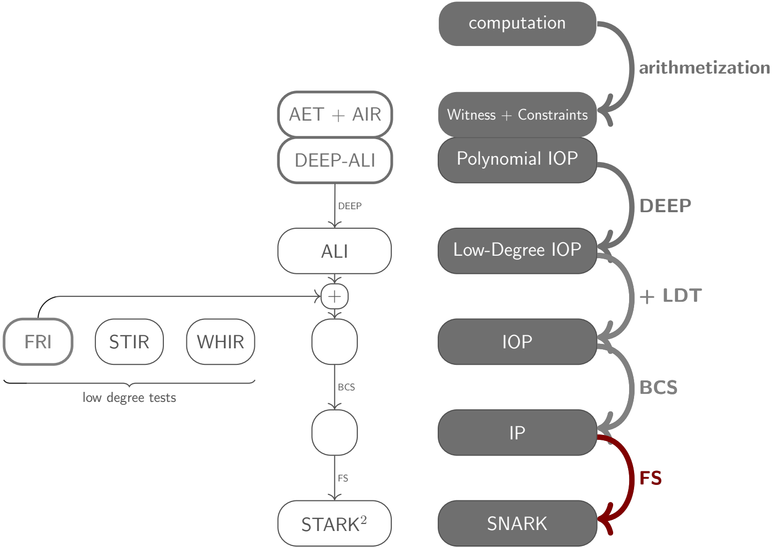
FRI $\{ \quad +(3 \cdot \mathsf{w} + 2 \cdot \#\mathcal{C} + \log N) \cdot \epsilon_{\mathsf{GAP}}(\delta_0) + (1 - \delta_0)^t$

BCS $\{ \quad + \frac{Q \cdot (Q-1)}{2^\lambda}$

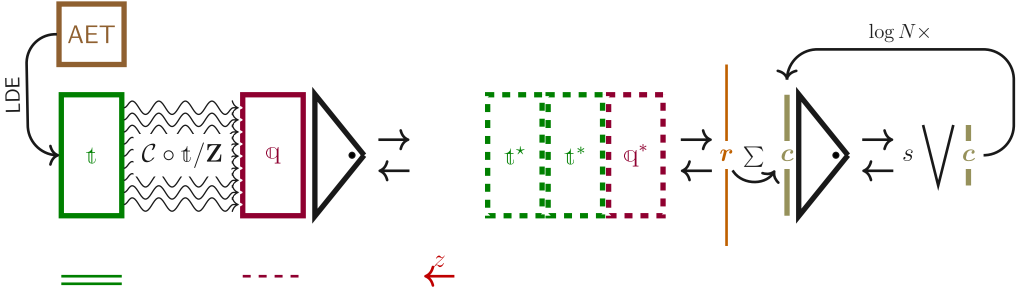
Table of Contents

Table of Contents

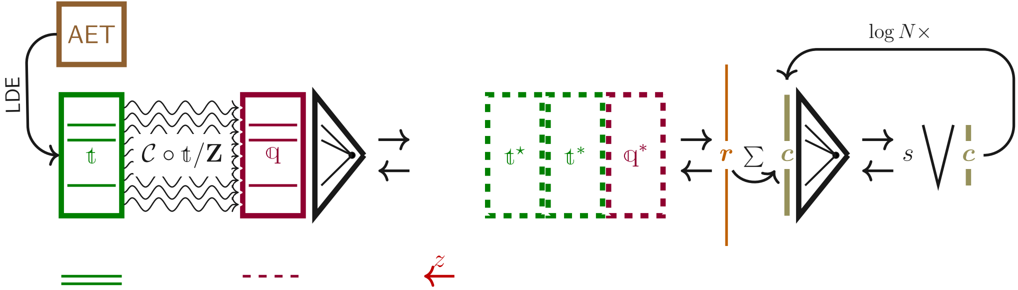
STARK Compilation Pipeline (Fiat-Shamir Transform)



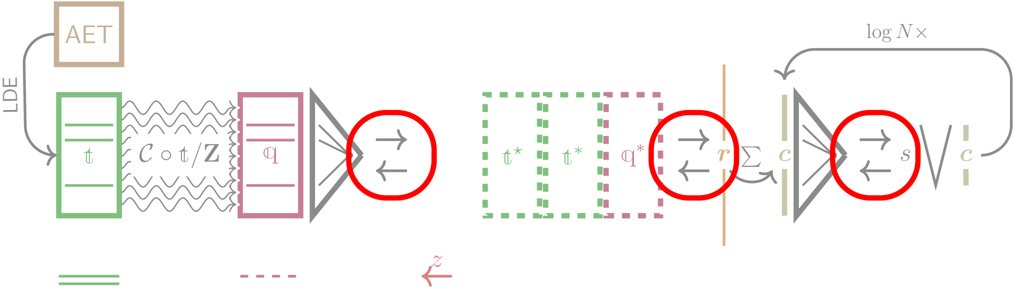
(Non-)Interactive STARK



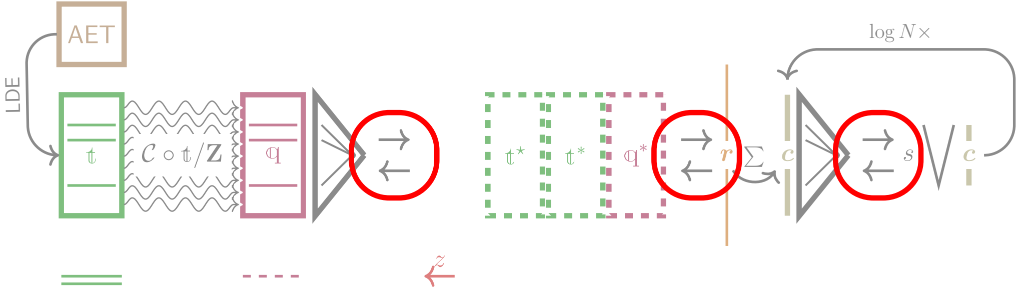
(Non-)Interactive STARK



(Non-)Interactive STARK



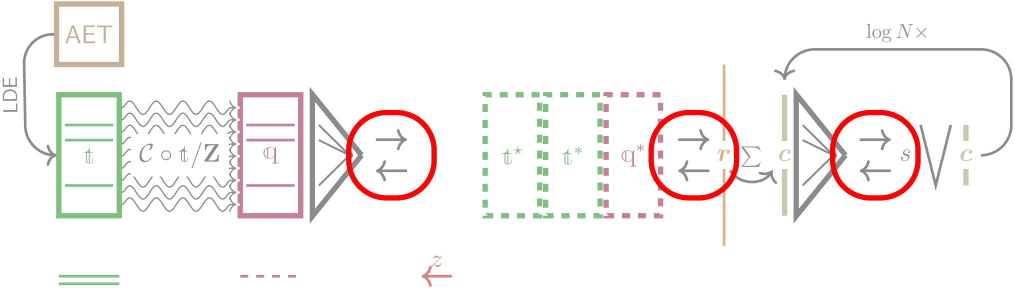
(Non-)Interactive STARK



Interactivity is a big problem in practice.

Can remove?

(Non-)Interactive STARK



Interactivity is a big problem in practice.

Can remove?

$\rightarrow$ Yes! But ...

IP to NIP

IP

random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\star$

Q evaluation queries

interaction

Prover Verifier

simulate

NIP

random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\star$

Q evaluation queries

~~interaction~~

Prover Verifier

IP to NIP

IP

random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\star$

Q evaluation queries

interaction

NIP

random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\star$

Q evaluation queries

~~interaction~~

simulate

Prover		Verifier
claim		claim
	$\xrightarrow{cmt}$	
	$\xleftarrow{c_1}$	
	$\xrightarrow{r_1}$	
	$\xleftarrow{c_2}$	
	$\xrightarrow{r_2}$	
$\vdots$	$\vdots$	$\vdots$

$\mathsf{V}(\text{claim}, cmt, c_1, r_1, c_2, r_2, \ldots)$

Prover	Verifier
claim	claim

IP to NIP

IP

random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^*$

Q evaluation queries

interaction

simulate

Prover		Verifier
claim		claim
	$\xrightarrow{cmt}$	
	$\xleftarrow{c_1}$	
	$\xrightarrow{r_1}$	
	$\xleftarrow{c_2}$	
	$\xrightarrow{r_2}$	
$\vdots$	$\vdots$	$\vdots$
		$\mathsf{V}(\text{claim}, cmt, c_1, r_1, c_2, r_2, \ldots)$

NIP

random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^*$

Q evaluation queries

~~interaction~~

Prover		Verifier
claim		claim
$\xrightarrow{cmt}$		
$\xleftarrow{c_1}$ $\mathsf{H}(\text{claim} \| cmt)$		
$\xrightarrow{r_1}$		
$\xleftarrow{c_2}$ $\mathsf{H}(\text{claim} \| cmt \| r_1)$		
$\vdots$		
	$\xrightarrow{cmt, r_1, r_2, \ldots}$	
		$c_1 \leftarrow \mathsf{H}(\text{claim} \| cmt)$
		$c_2 \leftarrow \mathsf{H}(\text{claim} \| cmt \| r_1)$
		$\vdots$
		$\mathsf{V}(\text{claim}, cmt, c_1, r_1, c_2, r_2, \ldots)$

IP to NIP

IP	NIP
random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\star$	random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\star$
Q evaluation queries	Q evaluation queries
interaction	~~interaction~~

simulate

IP

Prover		Verifier
claim		claim
	$\xrightarrow{cmt}$	
	$\xleftarrow{c_1}$	
	$\xrightarrow{r_1}$	
	$\xleftarrow{c_2}$	
	$\xrightarrow{r_2}$	
$\vdots$	$\vdots$	$\vdots$
		$\mathsf{V}(\text{claim}, cmt,$ $c_1, r_1, c_2, r_2, \ldots)$

NIP

Prover		Verifier
claim		claim
$\xrightarrow{cmt}$		
$\xleftarrow{c_1}$ $\mathsf{H}(\text{claim}\|cmt)$		
$\xrightarrow{r_1}$		
$\xleftarrow{c_2}$ $\mathsf{H}(\text{claim}\|cmt\|r_1)$		
$\vdots$		
	$\xrightarrow{cmt, r_1, r_2, \ldots}$	
		$c_1 \leftarrow \mathsf{H}(\text{claim}\|cmt)$
		$c_2 \leftarrow \mathsf{H}(\text{claim}\|cmt\|r_1)$
		$\vdots$
		$\mathsf{V}(\text{claim}, cmt,$ $c_1, r_1, c_2, r_2, \ldots)$

Fiat-Shamir Transform

Soundness of Fiat-Shamir: One Round

Task: find x such that $(x, \mathsf{H}(x)) \in \mathcal{R}$

for some *sparse* and *uniform* relation $\mathcal{R}$.

Soundness of Fiat-Shamir: One Round

Task: find x such that $(x, \mathsf{H}(x)) \in \mathcal{R}$

for some *sparse* and *uniform* relation $\mathcal{R}$.

sparse: $\epsilon_{\mathcal{R}} = \frac{|\{(x,y)\,|\,(x,y) \in \mathcal{R}\}|}{|\{(x,y)\}|}$ is small

uniform: $\forall x, y \,:\, |\{z \,|\, (x,z) \in \mathcal{R}\}| \approx |\{z \,|\, (y,z) \in \mathcal{R}\}|$

Soundness of Fiat-Shamir: One Round

Task: find x such that $(x, \mathsf{H}(x)) \in \mathcal{R}$

for some *sparse* and *uniform* relation $\mathcal{R}$.

sparse: $\epsilon_{\mathcal{R}} = \frac{|\{(x,y) \mid (x,y) \in \mathcal{R}\}|}{|\{(x,y)\}|}$ is small

uniform: $\forall x, y : |\{z \mid (x, z) \in \mathcal{R}\}| \approx |\{z \mid (y, z) \in \mathcal{R}\}|$

Question: what is the success probability?

Soundness of Fiat-Shamir: One Round

Task: find x such that $(x, \mathsf{H}(x)) \in \mathcal{R}$

for some *sparse* and *uniform* relation $\mathcal{R}$.

sparse: $\epsilon_{\mathcal{R}} = \frac{|\{(x,y) \mid (x,y) \in \mathcal{R}\}|}{|\{(x,y)\}|}$ is small

uniform: $\forall x, y \,:\, |\{z \mid (x, z) \in \mathcal{R}\}| \approx |\{z \mid (y, z) \in \mathcal{R}\}|$

Question: what is the success probability?

$\rightarrow$ depends on $\#$ queries

Soundness of Fiat-Shamir: One Round

Task: find x such that $(x, \mathsf{H}(x)) \in \mathcal{R}$

for some *sparse* and *uniform* relation $\mathcal{R}$.

sparse: $\epsilon_{\mathcal{R}} = \frac{|\{(x,y)\,|\,(x,y)\in\mathcal{R}\}|}{|\{(x,y)\}|}$ is small

uniform: $\forall x, y \, : \, |\{z \,|\, (x, z) \in \mathcal{R}\}| \approx |\{z \,|\, (y, z) \in \mathcal{R}\}|$

Question: what is the success probability?

$\rightarrow$ depends on $\#$ queries

– for 1 query: $\epsilon_{\mathcal{R}}$

– for Q queries: $Q \cdot \epsilon_{\mathcal{R}}$

Soundness of Fiat-Shamir: One Round

Task: find x such that $(x, \mathsf{H}(x)) \in \mathcal{R}$

for some *sparse* and *uniform* relation $\mathcal{R}$.

Apply to (Prover, Verifier):

Find cmt such that

$\mathsf{H}(\mathsf{claim} \| cmt) \in \mathcal{S}(cmt)$.

sparse: $\epsilon_{\mathcal{R}} = \frac{|\{(x,y) \,|\, (x,y) \in \mathcal{R}\}|}{|\{(x,y)\}|}$ is small

uniform: $\forall x, y \,:\, |\{z \,|\, (x, z) \in \mathcal{R}\}| \approx |\{z \,|\, (y, z) \in \mathcal{R}\}|$

Question: what is the success probability?

$\rightarrow$ depends on $\#$ queries

– for 1 query: $\epsilon_{\mathcal{R}}$

– for Q queries: $Q \cdot \epsilon_{\mathcal{R}}$

Soundness of Fiat-Shamir: One Round

Task: find x such that $(x, \mathsf{H}(x)) \in \mathcal{R}$

for some *sparse* and *uniform* relation $\mathcal{R}$.

sparse: $\epsilon_{\mathcal{R}} = \frac{|\{(x,y) \,|\, (x,y) \in \mathcal{R}\}|}{|\{(x,y)\}|}$ is small

uniform: $\forall x, y \,:\, |\{z \,|\, (x, z) \in \mathcal{R}\}| \approx |\{z \,|\, (y, z) \in \mathcal{R}\}|$

Apply to (Prover, Verifier):

Find cmt such that

$\mathsf{H}(\mathsf{claim} \| cmt) \in \mathcal{S}(cmt)$.

$\epsilon_V = \frac{|\mathcal{S}|}{2^\lambda} = \Pr[V \checkmark \,|\, \mathsf{claim} \times]$

$\checkmark\ |\mathcal{S}| \approx |\mathcal{S}(cmt)| \approx |\mathcal{S}(cmt')|$

Question: what is the success probability?

$\rightarrow$ depends on $\#$ queries

– for 1 query: $\epsilon_{\mathcal{R}}$

– for Q queries: $Q \cdot \epsilon_{\mathcal{R}}$

Soundness of Fiat-Shamir: One Round

Task: find x such that $(x, \mathsf{H}(x)) \in \mathcal{R}$

for some *sparse* and *uniform* relation $\mathcal{R}$.

sparse: $\epsilon_{\mathcal{R}} = \frac{|\{(x,y) \mid (x,y) \in \mathcal{R}\}|}{|\{(x,y)\}|}$ is small

uniform: $\forall x, y \, : \, |\{z \mid (x,z) \in \mathcal{R}\}| \approx |\{z \mid (y,z) \in \mathcal{R}\}|$

Question: what is the success probability?

$\rightarrow$ depends on # queries

– for 1 query: $\epsilon_{\mathcal{R}}$

– for Q queries: $Q \cdot \epsilon_{\mathcal{R}}$

Apply to (Prover, Verifier):

Find cmt such that

$\mathsf{H}(\mathsf{claim} \| cmt) \in \mathcal{S}(cmt)$.

$\epsilon_V = \frac{|\mathcal{S}|}{2^\lambda} = \Pr[V \checkmark \mid \mathsf{claim} \times]$

$\checkmark \; |\mathcal{S}| \approx |\mathcal{S}(cmt)| \approx |\mathcal{S}(cmt')|$

$\rightarrow$ soundness error?

$\rightarrow$ depends on # queries

$$Q \cdot \Pr[V \checkmark \mid \mathsf{claim} \times]$$

Intuition: Q attempts $\Rightarrow$ $Q\times \nearrow$ success prob

Soundness of Fiat-Shamir: Multiple Rounds

Find $(cmt, rsp_1, rsp_2, \ldots)$

such that $\mathsf{V}(\mathsf{claim}, cmt, ch_1, rsp_1, ch_2, rsp_2, \ldots) = 1$

where $ch_1 = \mathsf{H}(\mathsf{claim} \| cmt)$, $ch_2 = \mathsf{H}(\mathsf{claim} \| cmt \| rsp_1)$, ...

Soundness of Fiat-Shamir: Multiple Rounds

Find $(cmt, rsp_1, rsp_2, \ldots)$

such that $\mathsf{V}(\text{claim}, cmt, ch_1, rsp_1, ch_2, rsp_2, \ldots) = 1$

where $ch_1 = \mathsf{H}(\text{claim} \| cmt)$, $ch_2 = \mathsf{H}(\text{claim} \| cmt \| rsp_1)$, ...

$\mathcal{R}$ is well defined ✓

sparse ✓

uniformity $\times$

intuition: "good start" / "bad start"

Soundness of Fiat-Shamir: Round-by-Round Soundness

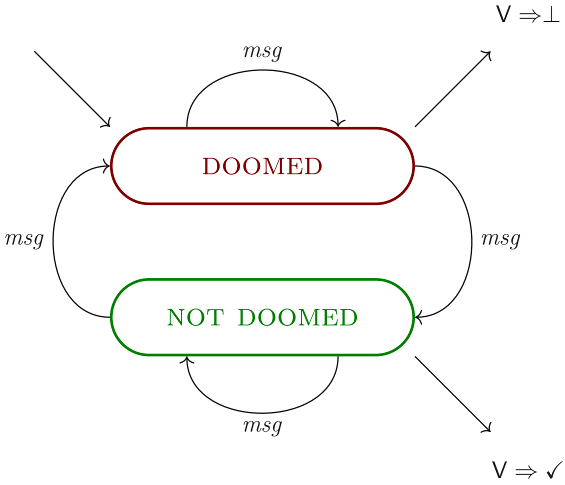
$\text{TRANSCRIPT PREFIXES} = \text{DOOMED} \sqcup \text{NOT DOOMED}$

DOOMED

NOT DOOMED

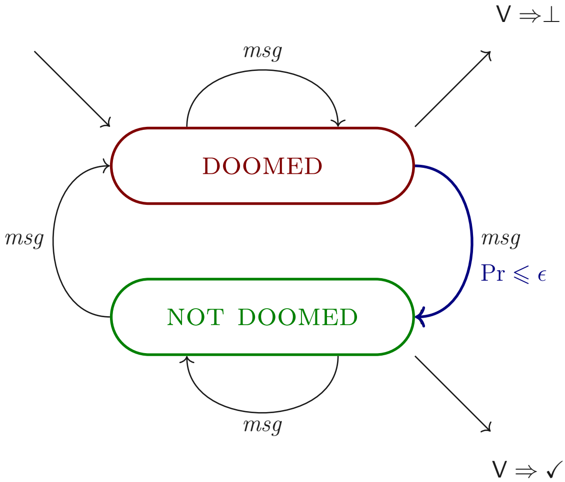
Soundness of Fiat-Shamir: Round-by-Round Soundness

$$\text{TRANSCRIPT PREFIXES} = \text{DOOMED} \sqcup \text{NOT DOOMED}$$



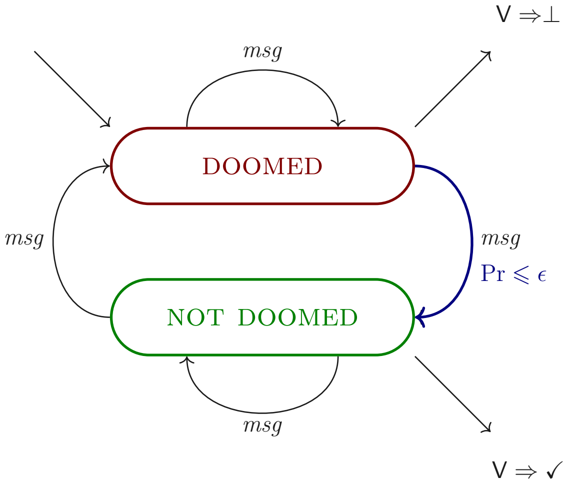
Soundness of Fiat-Shamir: Round-by-Round Soundness

$$\text{TRANSCRIPT PREFIXES} = \text{DOOMED} \sqcup \text{NOT DOOMED}$$



Soundness of Fiat-Shamir: Round-by-Round Soundness

$\textsc{transcript prefixes} = \textsc{doomed} \sqcup \textsc{not doomed}$



- RBR soundness ⇒ FS soundness
- RBR soundness of FRI established in [BGKTTZ23]
  - → requires *weighted* correlated agreement [BCIKS20]
- RBR soundness of DEEP-ALI established ???
  - → specific example: EthSTARK ✓

Table of Contents

Table of Contents

Table of Contents

Preview

Next Lecture:

- Optimizations
  - Quotient Segments
  - Univariate and Multilinear Batching
  - Grinding
- Enhancements
  - Zero-Knowledge
  - Randomized AIR (without Preprocessing)
- VM Architecture
  - Example / Overview
  - Communication Arguments
  - Memory
- Other Topics

Table of Contents

Table of Contents

Introduction to STARKs

Alan Szepieniec
艾伦·佘丕涅茨
alan@neptune.cash

https://neptune.cash/

https://triton-vm.org/

https://asz.ink/presentations/2025-09-18-Introduction-to-STARKs.pdf