

Introduction to STARKs

Alan Szepieniec

艾伦·余丕涅茨

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

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Motivation

Virtual Machine



Fixed Circuit

Motivation

Virtual Machine



Fixed Circuit

— versatile

high one-time cost

low per-application cost

slower + more resources required

— optimizable

high per-application cost

faster + fewer resources required

Motivation

Virtual Machine



Fixed Circuit

- versatile

- high one-time cost

- low per-application cost

- slower + more resources required

- one prover, one verifier, many programs

- optimizable

- high per-application cost

- faster + fewer resources required

- one prover and one verifier *for each* application

Motivation

Virtual Machine



Fixed Circuit

- 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

- 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



Fixed Circuit

- 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

- 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



Fixed Circuit

- 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

- 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

based on
STARK?

zkVM	ISA	team	open source		mainnet capable	Ethproofs
			verifier	GPU prover		
Pico	RISC-V	Brevis	✓✓ dual	ETA: soon™	✓	✓
SP1	RISC-V	Succinct	✓✓ dual	binaries only	✓	✓
Zirex	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

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

- Overview

- Arithmetization

- DEEP-ALI

- DEEP

- Low Degree Testing

- BCS Transform

- Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Definition

Scalable, Transparent **AR**gument of **K**nowledge

STARK¹: any proof system that is

- *transparent* $\Rightarrow$ no trusted setup
- *succinct* $\Rightarrow O(\text{poly } \log n)$ verifier
- *scalable* $\Rightarrow \text{succinct} + O(n \log n)$ 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, Transparent **AR**gument of **K**nowledge

STARK¹: any proof system that is

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

STARK²: 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, Transparent **AR**gument of **K**nowledge

STARK¹: any proof system that is

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

STARK²: a specific family of proof systems with

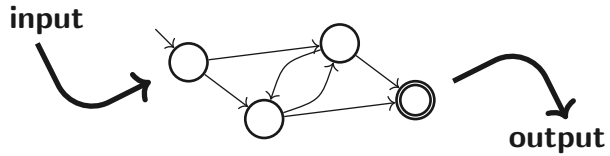
- *algebraic execution trace (AET)* and *algebraic intermediate representation (AIR)*
- *randomized AIR with preprocessing (RAP)*
- *ALI / DEEP-ALI*
- an *interactive oracle proof of proximity* such as *FRI / STIR / WHIR*
- *Merkle trees*

STARK³: a concrete *non-interactive* proof for some STARK²

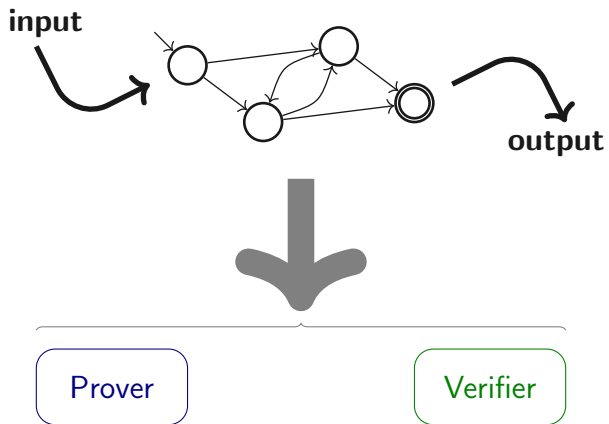
*in some unrealistic world, e.g., ROM or rewinding

Concept

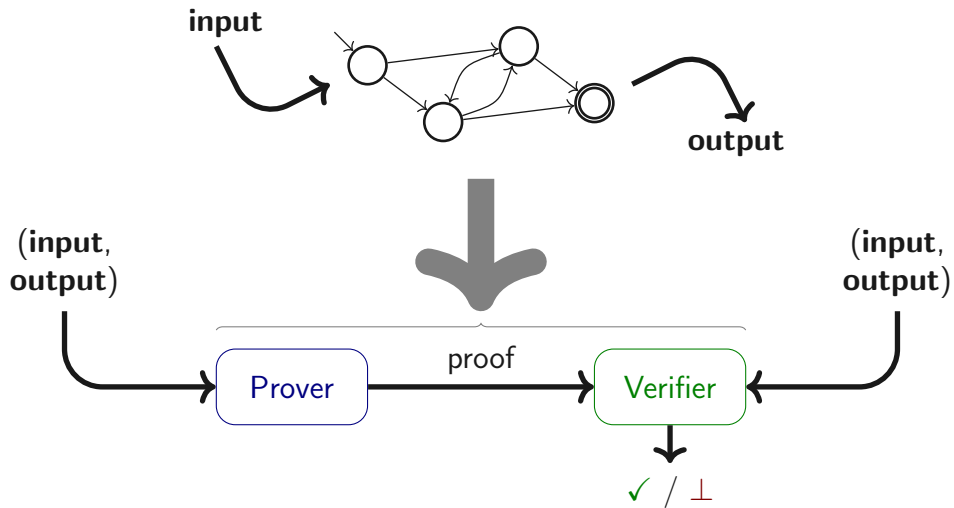
Concept



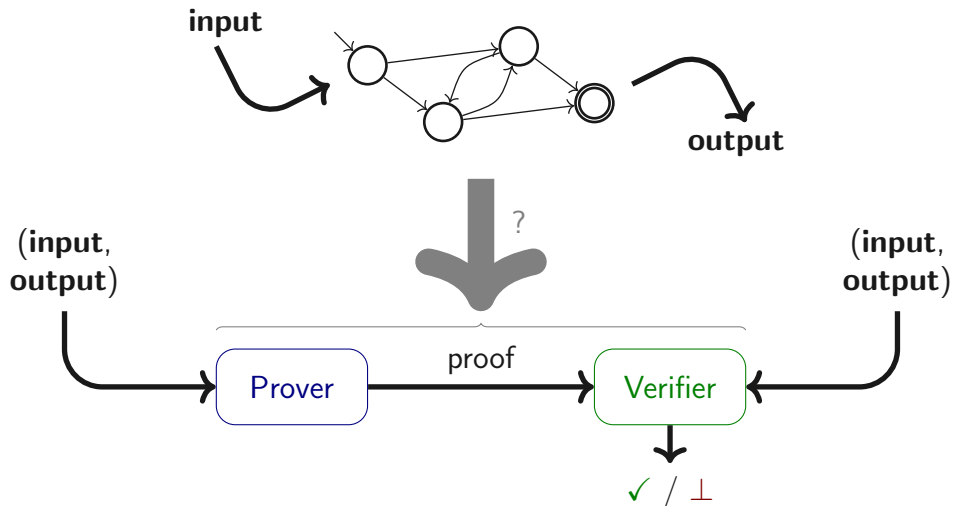
Concept



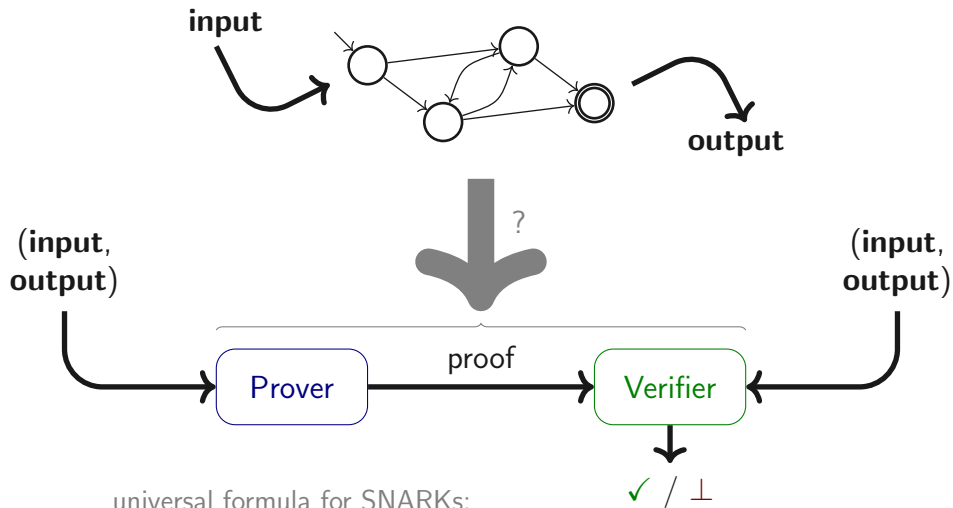
Concept



Concept



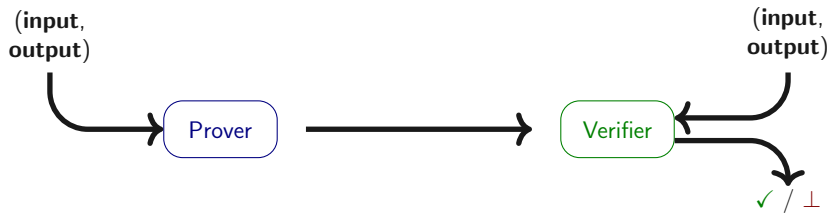
Concept



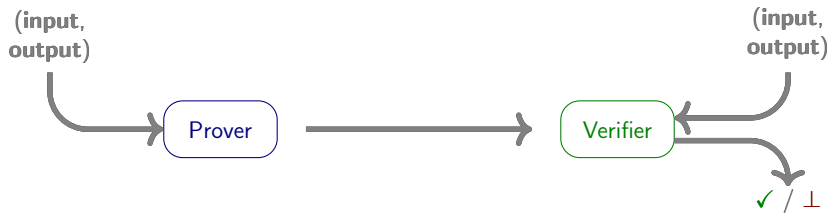
universal formula for SNARKs:

1. give Verifier superpowers
2. strip them away

Superpowers

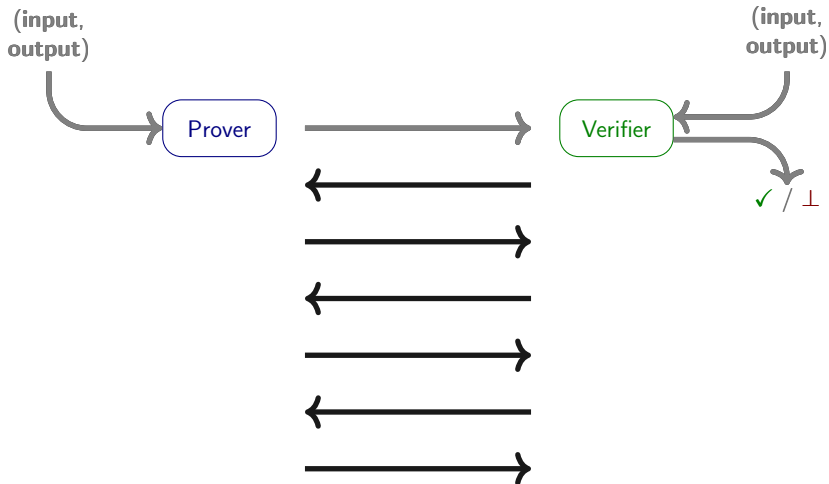


Superpowers



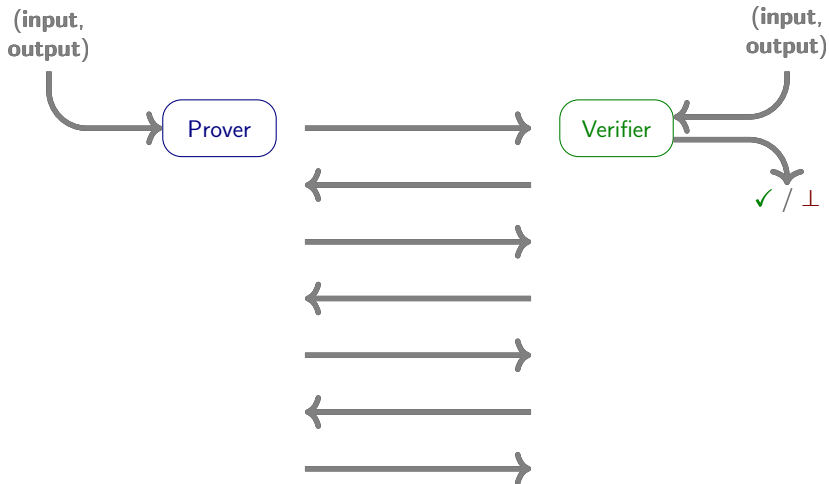
Superpowers

1. interaction



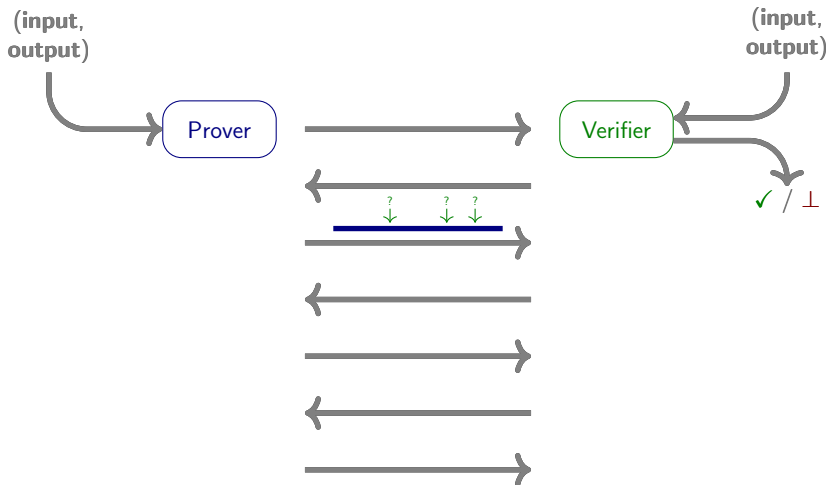
Superpowers

1. interaction



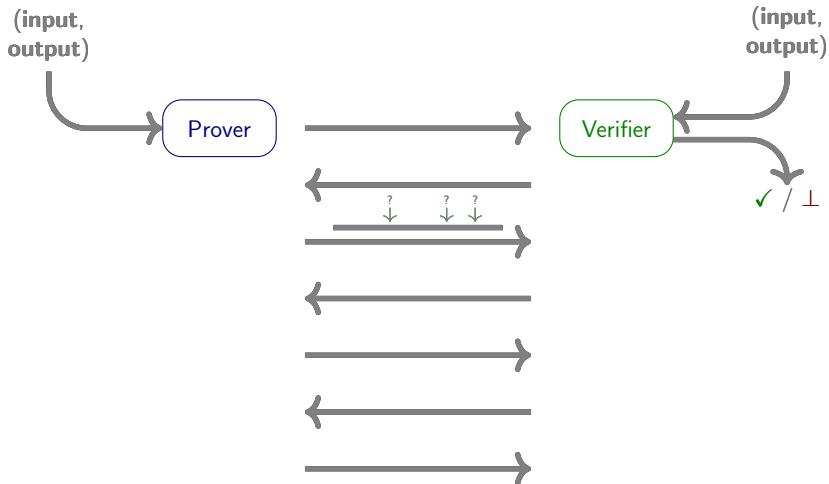
Superpowers

1. interaction
2. point queries



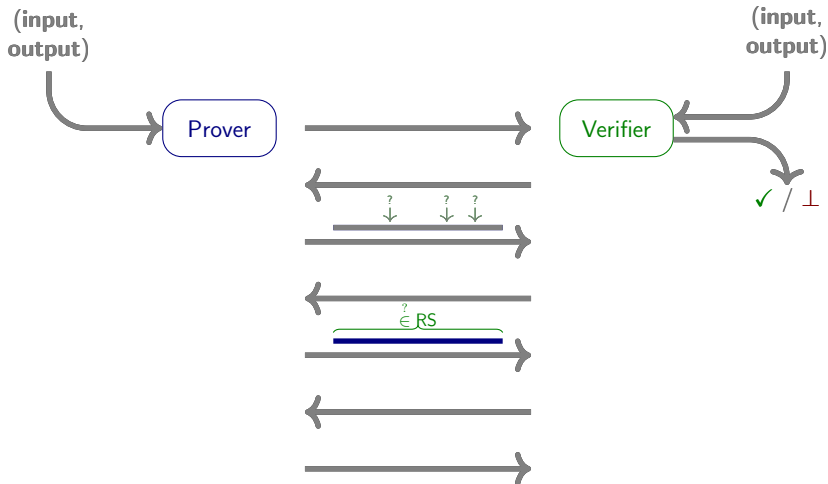
Superpowers

1. interaction
2. point queries



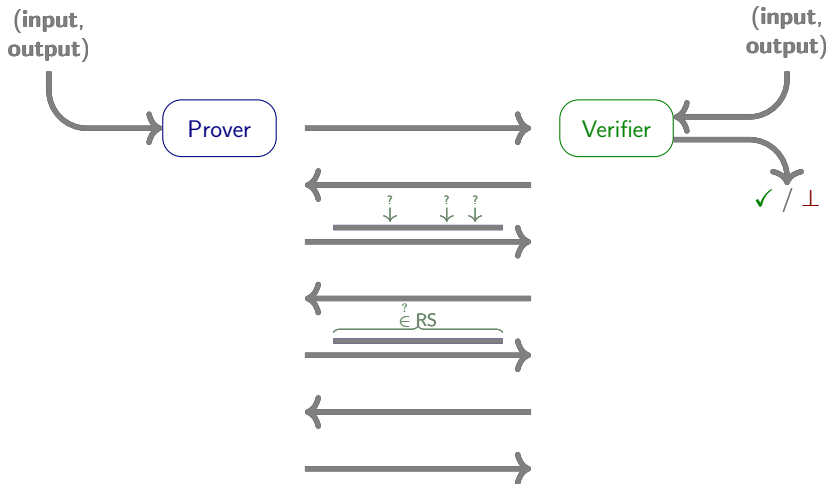
Superpowers

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



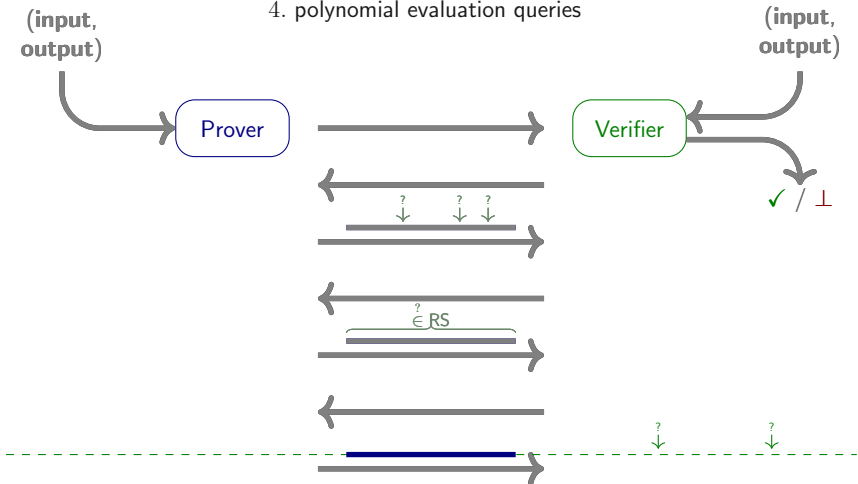
Superpowers

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



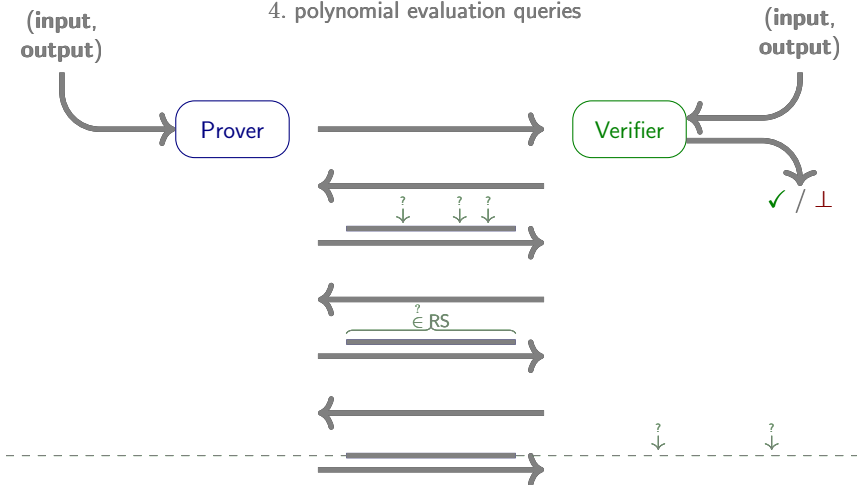
Superpowers

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

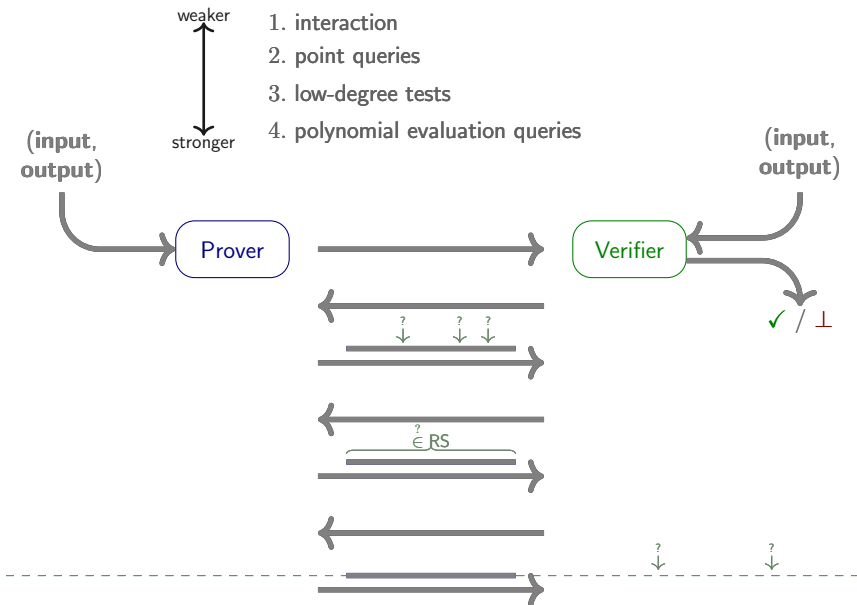


Superpowers

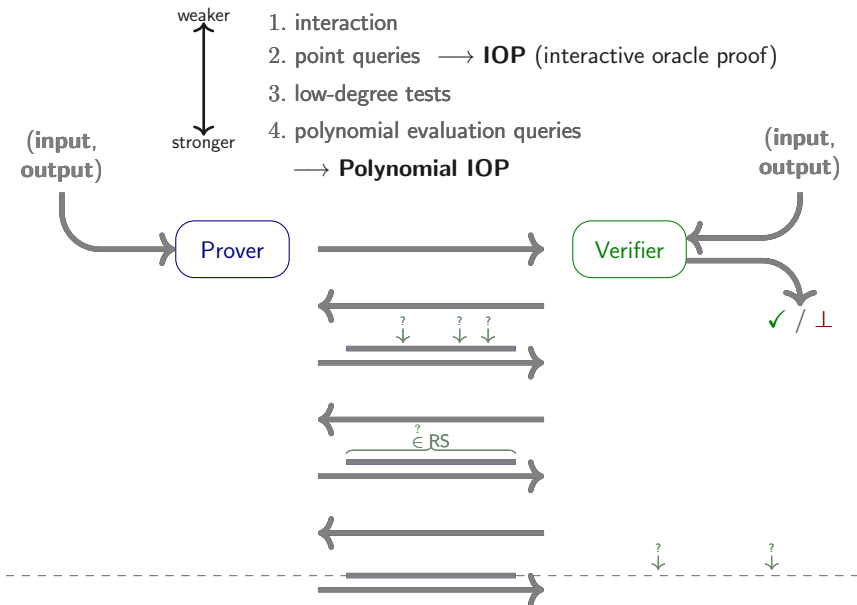
1. interaction
2. point queries
3. low-degree tests
4. polynomial evaluation queries



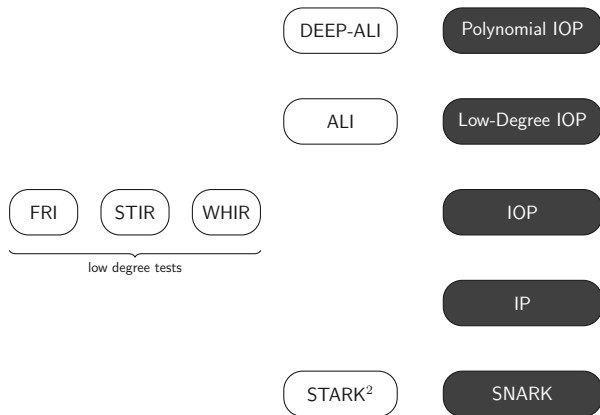
Superpowers



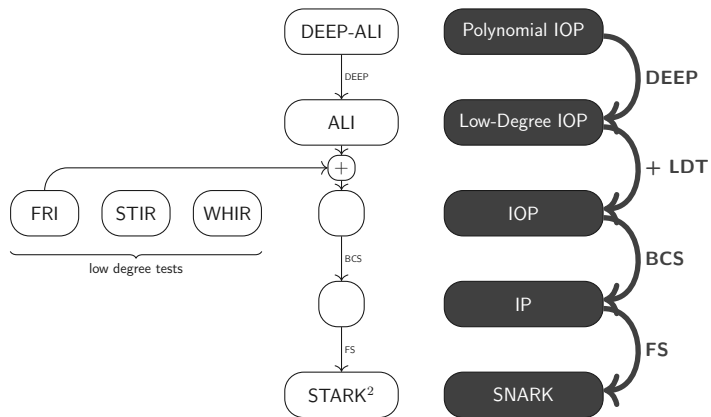
Superpowers



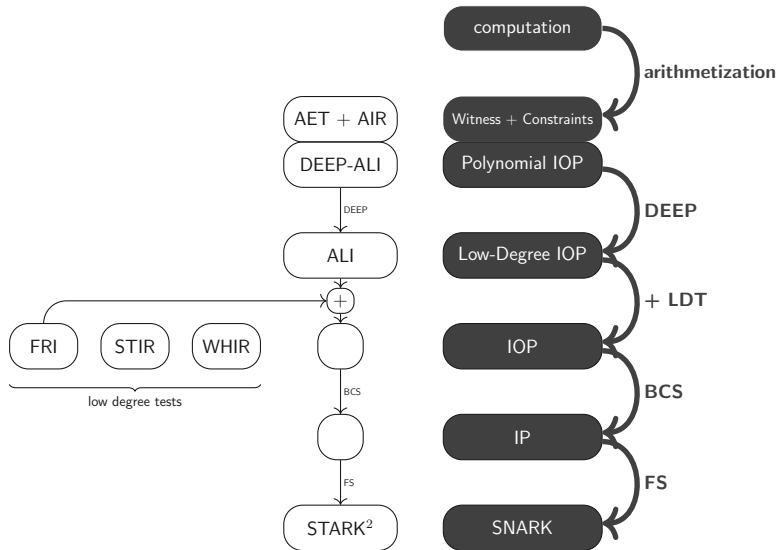
STARK Compilation Pipeline



STARK Compilation Pipeline



STARK Compilation Pipeline



STARK Compilation Pipeline

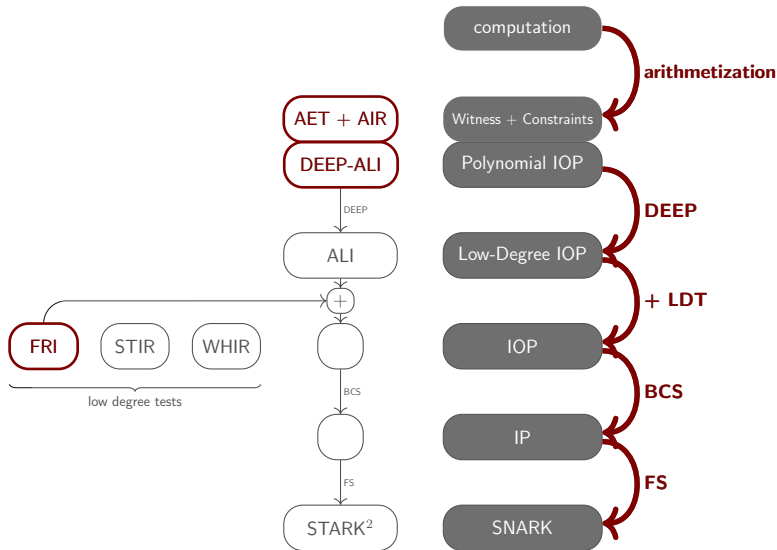


Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

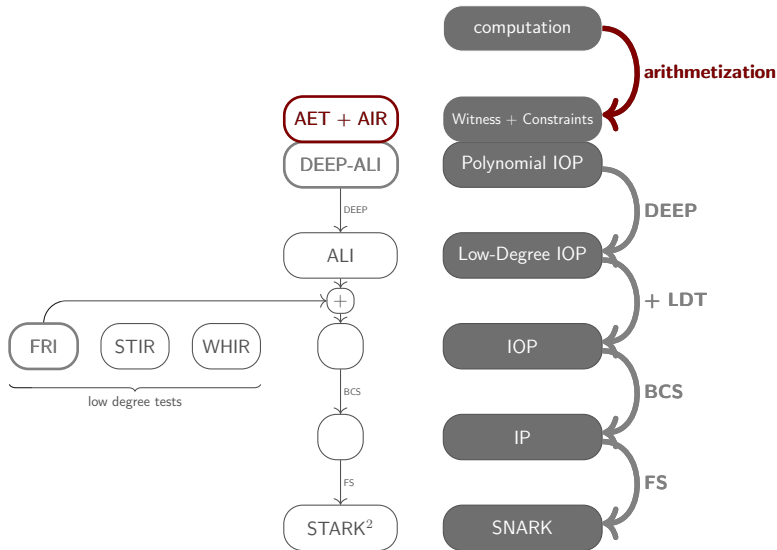
Low Degree Testing

BCS Transform

Fiat-Shamir Transform

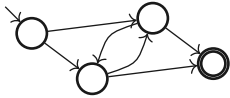
Preview

STARK Compilation Pipeline (Arithmetization)

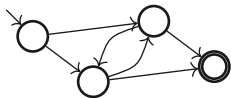


State Machine Arithmetization

State Machine Arithmetization



State Machine Arithmetization

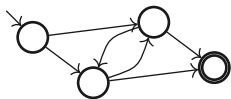


arithmetize:

describe in terms of

- finite field elements
- low degree polynomials over finite fields

State Machine Arithmetization



arithmetize:

describe in terms of

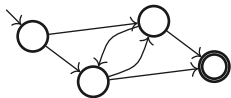
- finite field elements
- low degree polynomials over finite fields

state space:

$\mathbb{F}^w$

vectors of w finite field elements

State Machine Arithmetization



arithmetize:

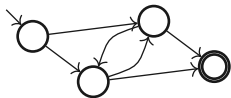
describe in terms of

- finite field elements
- low degree polynomials over finite fields

state space:

→ e.g. initial state $\mathbb{F}^w$ vectors of w finite field elements
 $(x_0, y_0) \in \mathbb{F}^2$

State Machine Arithmetization



arithmetize:

describe in terms of

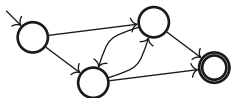
- finite field elements
- low degree polynomials over finite fields

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

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

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

State Machine Arithmetization



arithmetize:

describe in terms of

- finite field elements
- low degree polynomials over finite fields

state space:

→ e.g. initial state

$\mathbb{F}^w$ *vectors of w finite field elements*
 $(x_0, y_0) \in \mathbb{F}^2$

transition function:

→ evolution

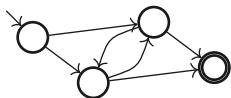
→ e.g.

$F : \mathbb{F}^w \rightarrow \mathbb{F}^w$ and $F(\mathbf{x}) \in (\mathbb{F}[\mathbf{x}]^{\leq d})^w$

$\mathbf{x}_0 \xrightarrow{F} \mathbf{x}_1 \xrightarrow{F} \mathbf{x}_2 \xrightarrow{F} \dots$

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

State Machine Arithmetization



arithmetize:

describe in terms of

- finite field elements
- low degree polynomials over finite fields

state space:

→ e.g. initial state

$\mathbb{F}^w$ *vectors of w finite field elements*
 $(x_0, y_0) \in \mathbb{F}^2$

transition function:

→ evolution

→ e.g.

$F : \mathbb{F}^w \rightarrow \mathbb{F}^w$ and $F(\mathbf{x}) \in (\mathbb{F}[\mathbf{x}]^{\leq d})^w$
 $\mathbf{x}_0 \xrightarrow{F} \mathbf{x}_1 \xrightarrow{F} \mathbf{x}_2 \xrightarrow{F} \dots$

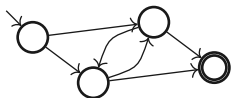
$$F(x, y) = \begin{pmatrix} 1 + xy \\ x^2 + y^2 \end{pmatrix}$$

computational claim: *boundary constraints*

→ 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:

→ e.g. initial state

$\mathbb{F}^w$ vectors of w finite field elements
 $(x_0, y_0) \in \mathbb{F}^2$

transition function:

→ evolution

→ e.g.

$F : \mathbb{F}^w \rightarrow \mathbb{F}^w$ and $F(\mathbf{x}) \in (\mathbb{F}[\mathbf{x}]^{\leq d})^w$
 $\mathbf{x}_0 \xrightarrow{F} \mathbf{x}_1 \xrightarrow{F} \mathbf{x}_2 \xrightarrow{F} \dots$

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

computational claim: boundary constraints

→ e.g.

$$\{x_0 = 1, y_0 = 2, x_7 = 3\}$$

integrity:

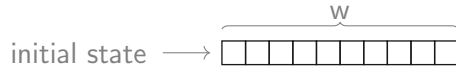
1. boundary constraints ✓

2. transition constraints ✓

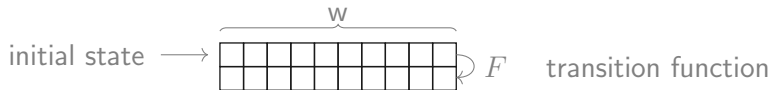
$$\Leftrightarrow \forall i \mathbf{x}_{i+1} = F(\mathbf{x}_i)$$

Algebraic Execution Trace (AET)

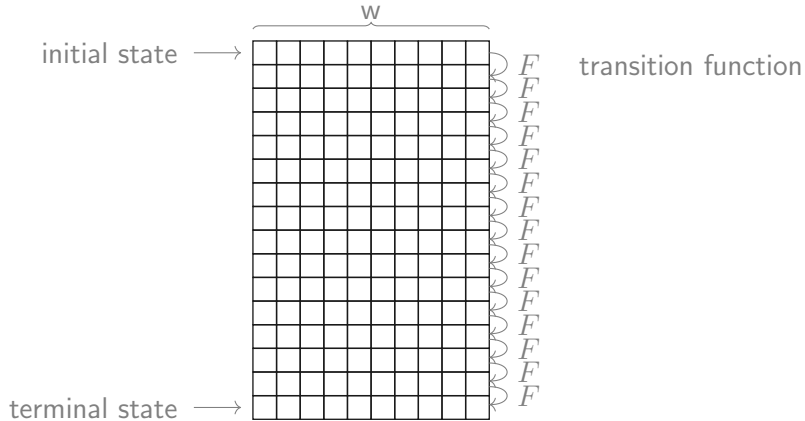
Algebraic Execution Trace (AET)



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:

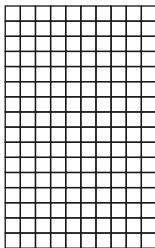
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint

type

AIR

AIR:

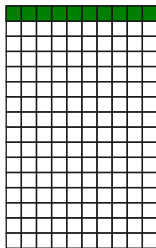
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint

$$\mathbf{x}_0 - \mathbf{x}_{init} = 0$$

type

initial

AIR

AIR:

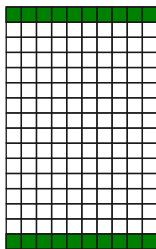
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint

$$\mathbf{x}_0 - \mathbf{x}_{init} = 0$$

type

initial

$$a(\mathbf{x}_{N-1}) = 0$$

terminal

AIR

AIR:

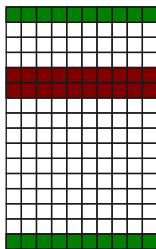
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint

$$\mathbf{x}_0 - \mathbf{x}_{init} = 0$$

$$\mathbf{x}_{i+1} - F(\mathbf{x}_i) = 0$$

$$a(\mathbf{x}_{N-1}) = 0$$

type

initial

transition

terminal

AIR

AIR:

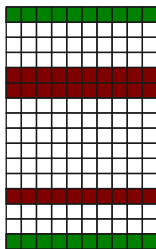
set of *multivariate* polynomials

of *low degree*

in a *combination* of AET rows

evaluates to zero $\Leftrightarrow$ AET is integral

AET



AIR constraint

$$\mathbf{x}_0 - \mathbf{x}_{init} = 0$$

$$\mathbf{x}_{i+1} - F(\mathbf{x}_i) = 0$$

$$c(\mathbf{x}_j) = 0$$

$$a(\mathbf{x}_{N-1}) = 0$$

type

initial

transition

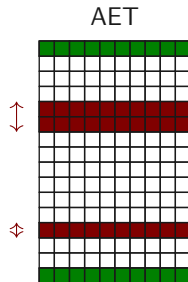
consistency

terminal

AIR

AIR:

set of *multivariate* polynomials
of *low degree*
in a *combination* of AET rows
evaluates to zero $\Leftrightarrow$ AET is integral



AIR constraint

$$\mathbf{x}_0 - \mathbf{x}_{init} = 0$$

$$\mathbf{x}_{i+1} - F(\mathbf{x}_i) = 0$$

$$c(\mathbf{x}_j) = 0$$

$$a(\mathbf{x}_{N-1}) = 0$$

type

initial

transition

consistency

terminal

unquantified

— applies to fixed row (combination)

quantified

— applies for all row combinations

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

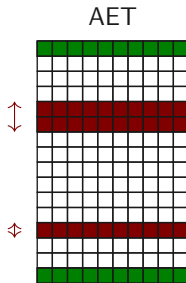
$$\forall j \in \{0, \dots, N-1\} : c(\mathbf{x}_j) = 0$$

AIR

AIR:

set of *multivariate* polynomials
of *low degree*

in a *combination* of AET rows
evaluates to zero $\Leftrightarrow$ AET is integral



AIR constraint

$$\mathbf{x}_0 - \mathbf{x}_{init} = 0$$

$$\mathbf{x}_{i+1} - F(\mathbf{x}_i) = 0$$

$$c(\mathbf{x}_j) = 0$$

$$a(\mathbf{x}_{N-1}) = 0$$

type

initial

transition

consistency

terminal

unquantified

— applies to fixed row (combination)

quantified

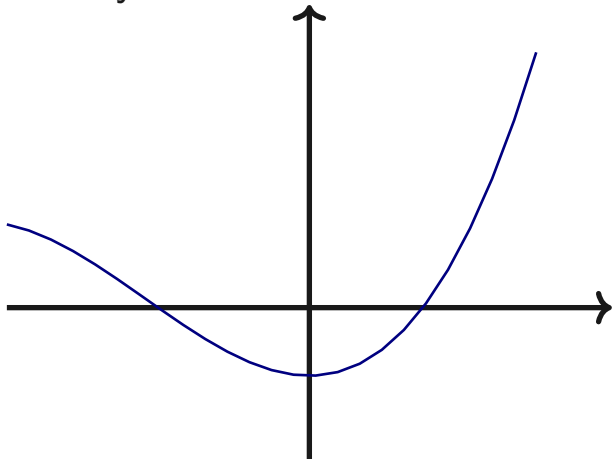
— applies for all row combinations

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

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

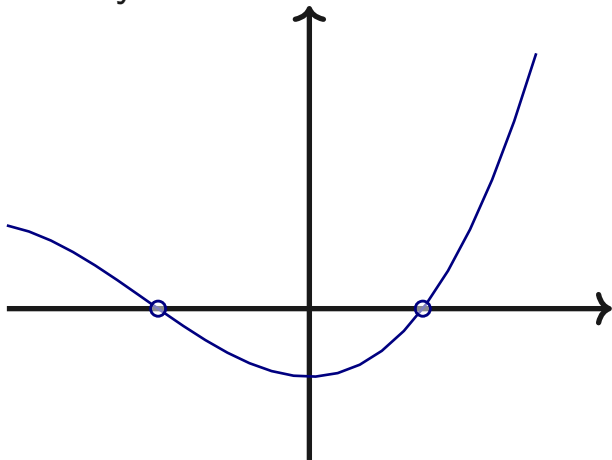
Q: How to *succinctly* test *quantified* constraints?

Divisibility



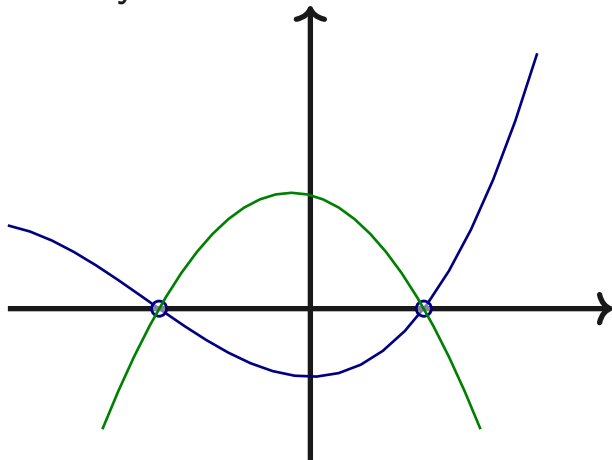
1. Given a polynomial $f(X)$

Divisibility



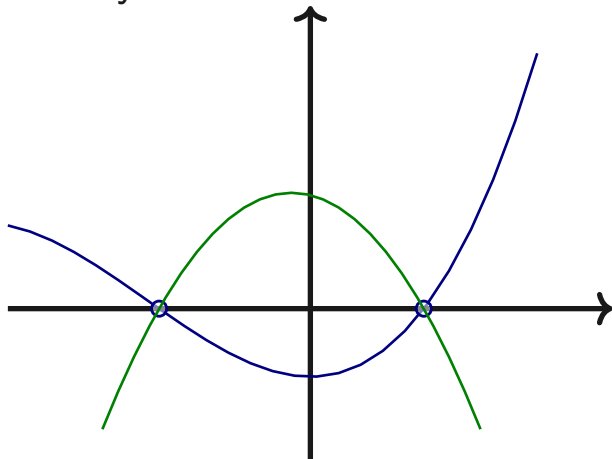
1. Given a polynomial $f(X)$
with zeros in $\{x_0, x_1, \dots\}$

Divisibility



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

Divisibility



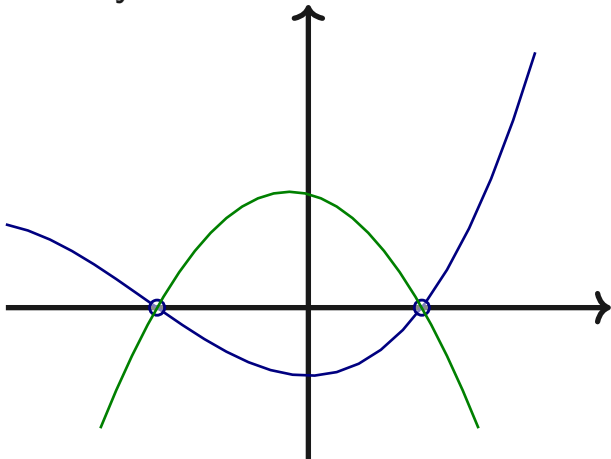
1. Given a polynomial $f(X)$
with zeros in $\{x_0, x_1, \dots\}$

2. Consider the *zeroifier*²

$$Z(X) = \prod_i (X - x_i)$$

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

Divisibility



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

2. Consider the *zeroifier*²

$$Z(X) = \prod_i (X - x_i)$$

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

Proof.

$$\text{Let } f(X) = k(X) \cdot Z(X) + r(X)$$

with $\deg(r) < \deg(Z)$.

$$\text{Then } \forall i \ f(x_i) = k(x_i) \cdot Z(x_i) + r(x_i)$$

$$0 = r(x_i).$$

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

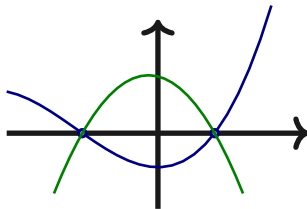
Therefore, $r(X) = 0$. $\square$

²also known as "vanishing polynomial"

Use Divisibility as Quantifier



quantifier



zero-finder

Q: how to *succinctly* test
quantified constraints?

A: express as *polynomial divisibility*
relation.

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

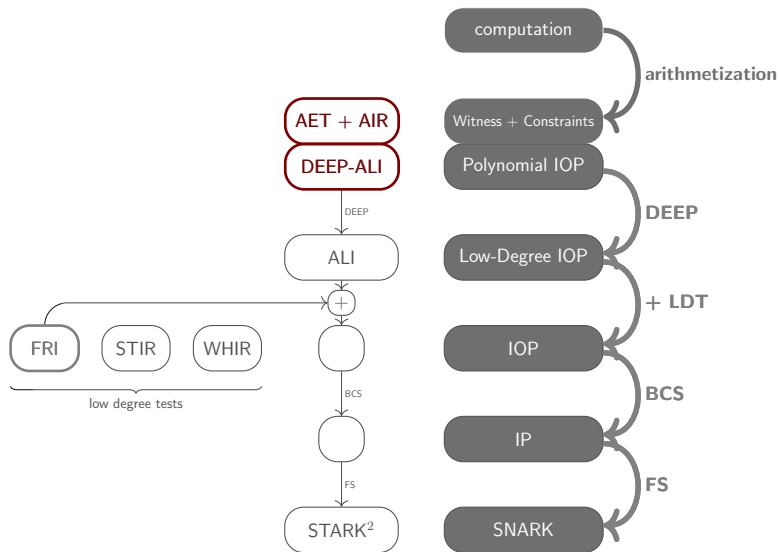
Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

STARK Compilation Pipeline (DEEP-ALI)



DEEP-AI: Intuition

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

DEEP-ALL: Intuition

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

Prover
trace T

Verifier
AIR $\mathcal{C}$

DEEP-ALL: Intuition

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

Prover
trace T

Verifier
AIR $\mathcal{C}$

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

$t(X)$



DEEP-AI: Intuition

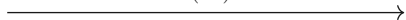
1. **Interpolate** trace polynomials
2. **Compose** with AIR
3. **Divide** out zerofiers

Prover
trace T

Verifier
AIR $\mathcal{C}$

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

$t(X)$



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

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

DEEP-AI: Intuition

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

Problems:

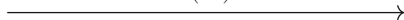
1. Soundness?
2. transition constraints $c(\mathbf{x}_i, \mathbf{x}_{i+1})$?
3. Polynomial IOP:
 - evaluation queries ✓
 - divisibility checks ✗

Prover
trace T

Verifier
AIR $\mathcal{C}$

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

$t(X)$



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

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

DEEP-AI: Intuition

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

Problems:

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

3. Polynomial IOP:

- evaluation queries ✓
- divisibility checks ✗

Prover
trace T

Verifier
AIR $\mathcal{C}$

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

$t(X)$



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

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

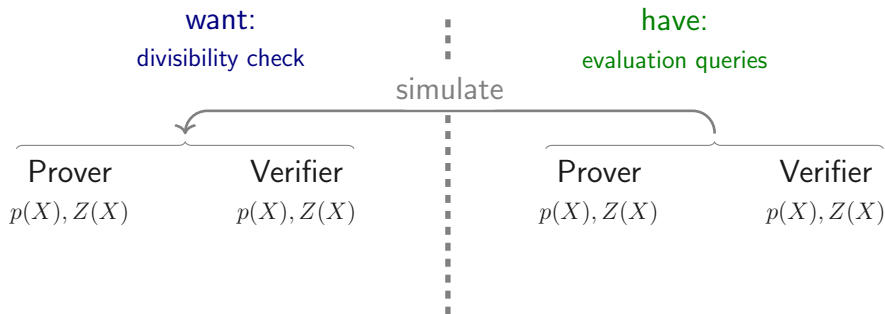
Polynomial IOP for Divisibility

want:
divisibility check

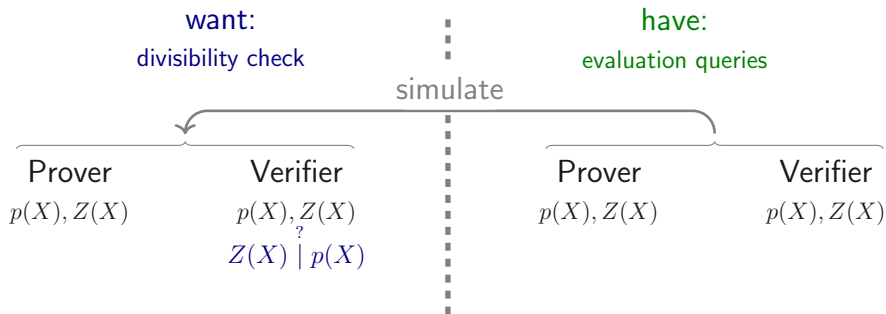


have:
evaluation queries

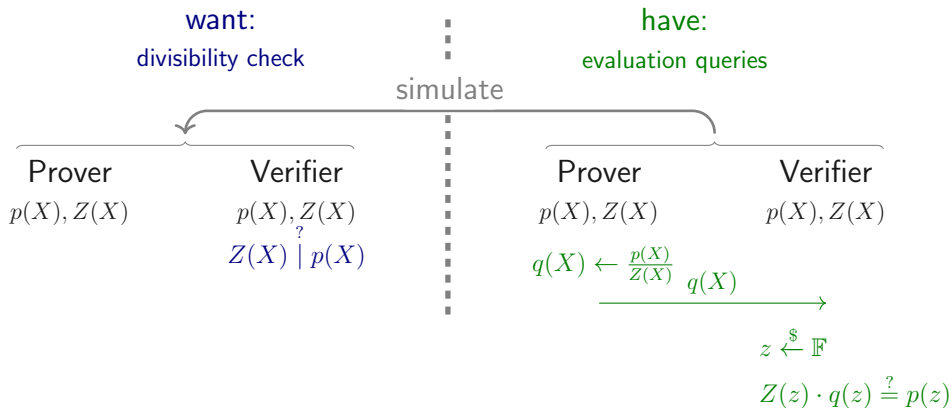
Polynomial IOP for Divisibility



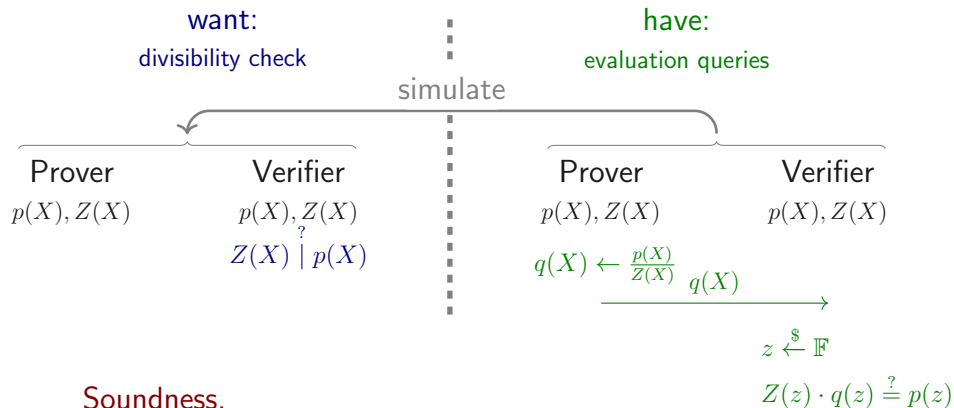
Polynomial IOP for Divisibility



Polynomial IOP for Divisibility



Polynomial IOP for Divisibility



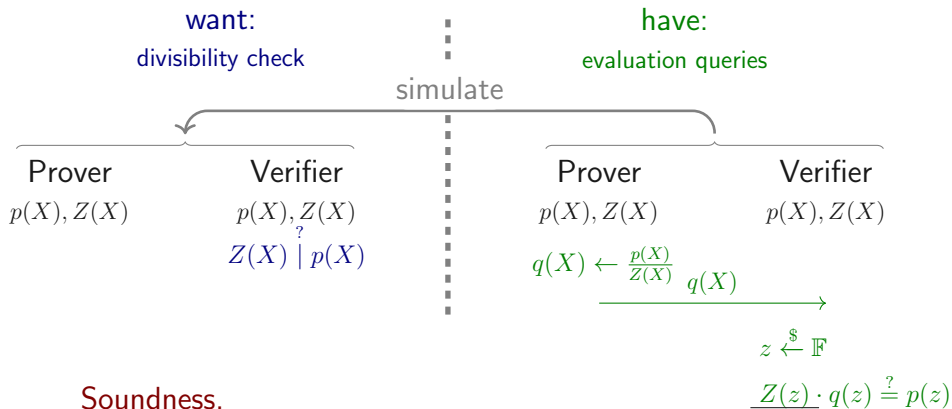
Soundness.

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

$$\leq \frac{\deg}{|\mathbb{F}|} \quad (\text{Schwartz-Zippel})$$

$$= \varepsilon.$$

Polynomial IOP for Divisibility



Soundness.

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

$$\leq \frac{\deg}{|\mathbb{F}|} \quad (\text{Schwartz-Zippel})$$

$$= \varepsilon.$$

evaluate efficiently?

→ sparsity

DEEP-AI: Refinement

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

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

Verifier
AIR $\mathcal{C}$

$t(X) \leftarrow \text{interpolate } T$
 $t(X)$
→

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

$q_j(X) \leftarrow \frac{c \circ t(X)}{Z(X)}$
 $q(X)$
→ $z \xleftarrow{\$} \mathbb{F}$

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

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

DEEP-AI: Refinement

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

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

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

$t(X)$

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

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

$q(X)$

Verifier
AIR $\mathcal{C}$

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

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

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

Problems:

- ~~1. Soundness? $\varepsilon = \frac{\deg}{|\mathbb{F}|}$ (1.a) what is deg?)~~
- ~~2. transition constraints $c(\mathbf{x}_i, \mathbf{x}_{i+1})$?~~
- ~~3. Polynomial IOP:
– evaluation queries $\checkmark$
– divisibility checks $\times$~~
4. sparse zerofier

DEEP-AI: Refinement

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

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

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

$t(X)$

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

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

$q(X)$

Verifier
AIR $\mathcal{C}$

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

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

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

Problems:

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

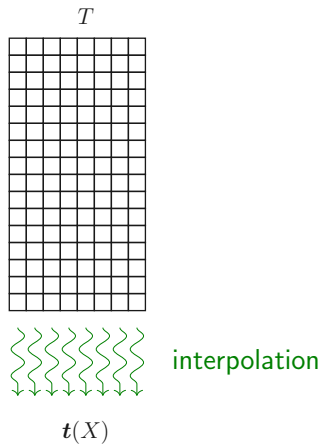
2. transition constraints $c(\mathbf{x}_i, \mathbf{x}_{i+1})?$

~~3. Polynomial IOP:~~

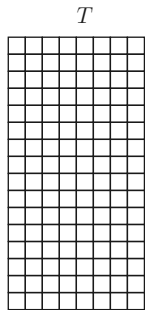
- ~~– evaluation queries $\checkmark$~~
- ~~– divisibility checks $\times$~~

4. sparse zerofier

Trace Interpolation Domain



Trace Interpolation Domain

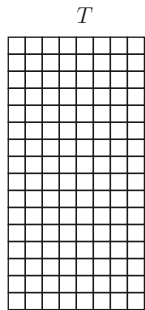


interpolation

Q: over which domain?

$t(X)$

Trace Interpolation Domain



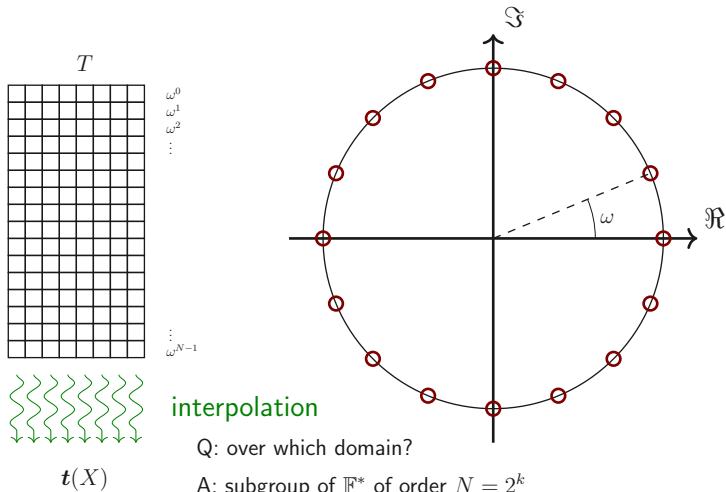
interpolation

$t(X)$

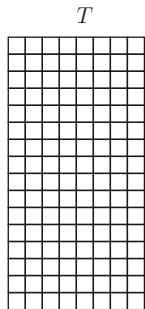
Q: over which domain?

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

Trace Interpolation Domain



Trace Interpolation Domain



ω^0
 ω^1
 ω^2
 $\vdots$

$\vdots$
 ω^{N-1}

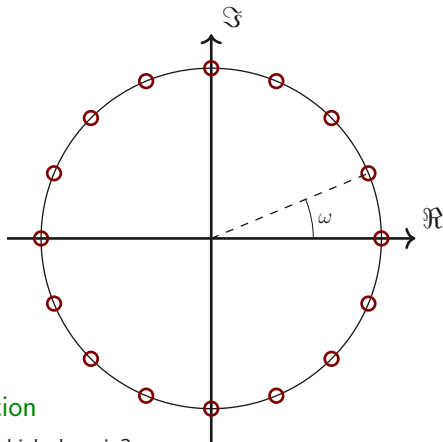


$t(X)$

interpolation

Q: over which domain?

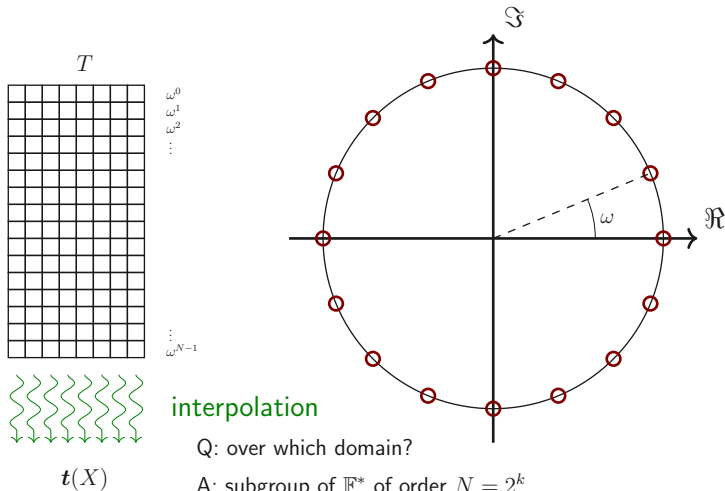
A: subgroup of $\mathbb{F}^*$ of order $N = 2^k$



★ complexity:

$O(N \log N)$ (NTT)

Trace Interpolation Domain



★ complexity:

$O(N \log N)$ (NTT)

★ zefiers:

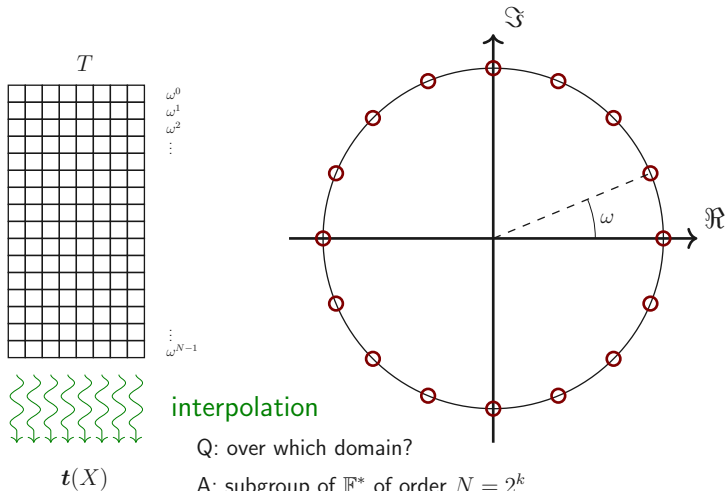
– first row: $X - 1$

– last row: $X - \omega^{-1}$

– entire domain: $X^N - 1$

– ... except for last row: $\frac{X^N - 1}{X - \omega^{-1}}$

Trace Interpolation Domain



★ complexity:

$$O(N \log N) \text{ (NTT)}$$

★ zeroifiers:

– 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

$t(\omega X)$ — rotation by 1 row

$(t(X), t(\omega X))$ — consecutive pairs

$$c(x_i, x_{i+1}) \circ t(X) \triangleq c(t(X), t(\omega X))$$

DEEP-ALI

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

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

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

$t(X)$

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

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

$q(X)$

Verifier
AIR $\mathcal{C}$

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

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

Problems:

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

~~2. transition constraints $c(x_i, x_{i+1})$?~~

~~3. Polynomial IOP:~~

- evaluation queries ✓
- divisibility checks ✗

~~4. sparse zerofier~~

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

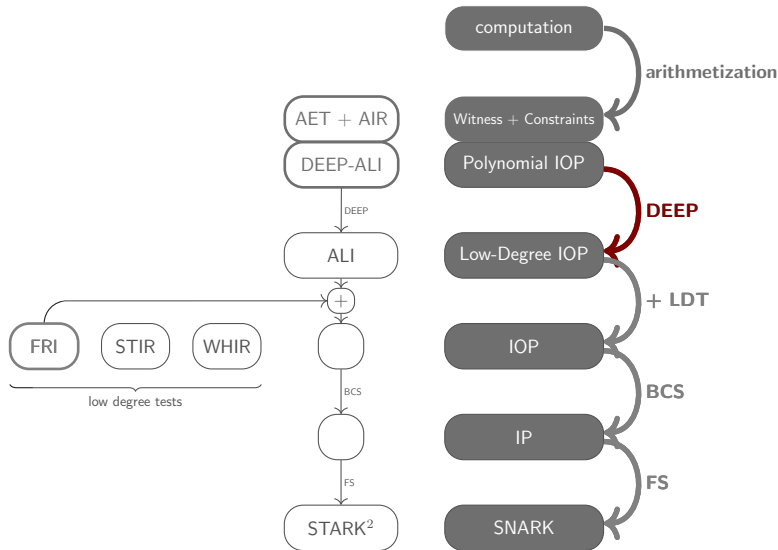
Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

STARK Compilation Pipeline (DEEP)



Polynomial IOP to Low-Degree IOP (1)

Polynomial IOP:

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

evaluation queries



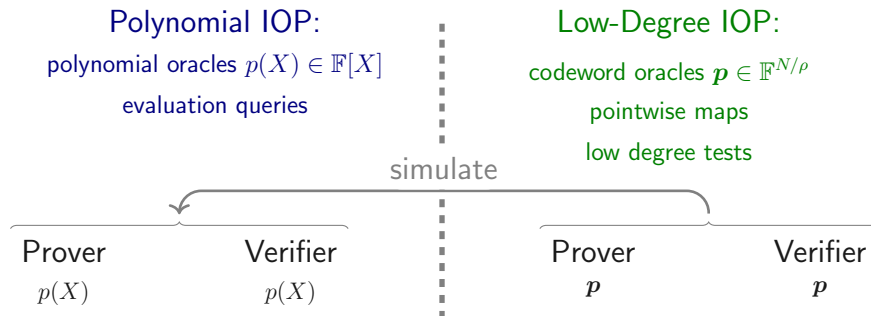
Low-Degree IOP:

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

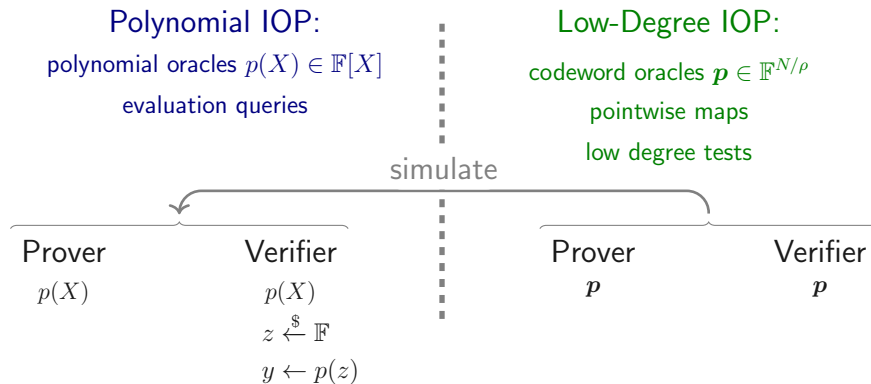
pointwise maps

low degree tests

Polynomial IOP to Low-Degree IOP (1)



Polynomial IOP to Low-Degree IOP (1)



Reed-Solomon Code

polynomial $\mapsto$ codeword
 $p(X)$ p

Reed-Solomon Code

polynomial $\mapsto$ codeword
 $p(X)$ $\mathbf{p}$

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

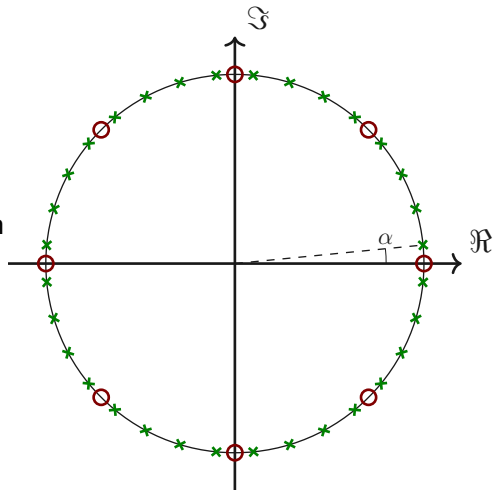
Reed-Solomon Code in STARKs

polynomial $\mapsto$ codeword
 $p(X)$ $\mathbf{p}$

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

○ trace *interpolation* domain

× trace *evaluation* domain



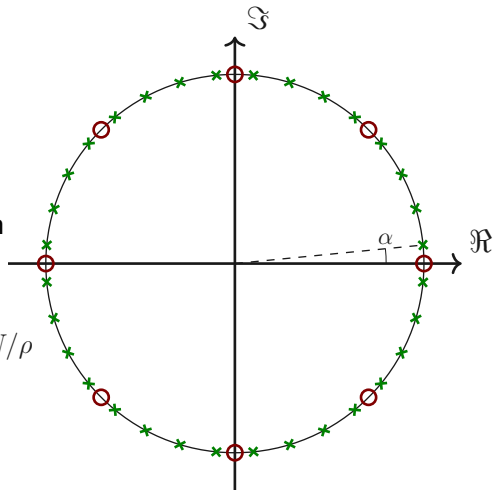
Reed-Solomon Code in STARKs

polynomial $\mapsto$ codeword
 $p(X)$ $\mathbf{p}$

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

○ trace *interpolation* domain

× trace *evaluation* domain
= coset of subgroup of order N/ρ



Reed-Solomon Code in STARKs

polynomial $\mapsto$ codeword
 $p(X)$ $\mathbf{p}$

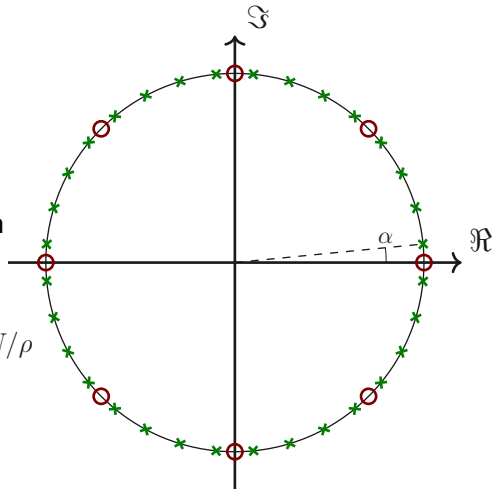
$$\mathbf{p} = p(D)$$

○ trace *interpolation* domain

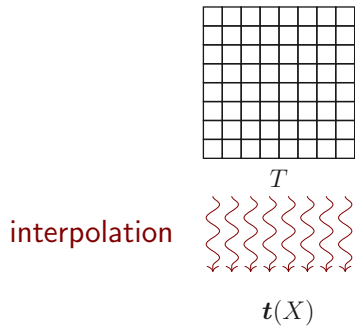
× trace *evaluation* domain
= coset of subgroup of order N/ρ

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

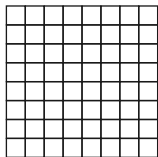
limits deg



Low-Degree Extension



Low-Degree Extension



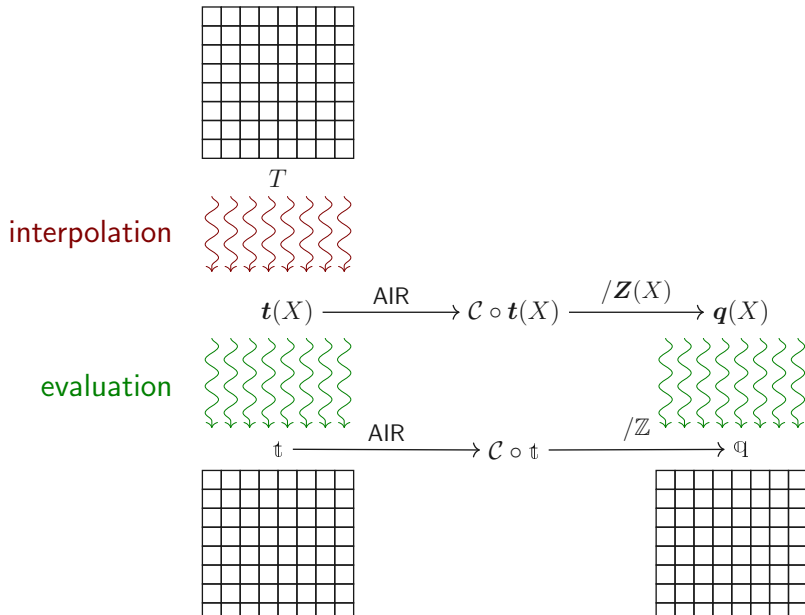
T

interpolation

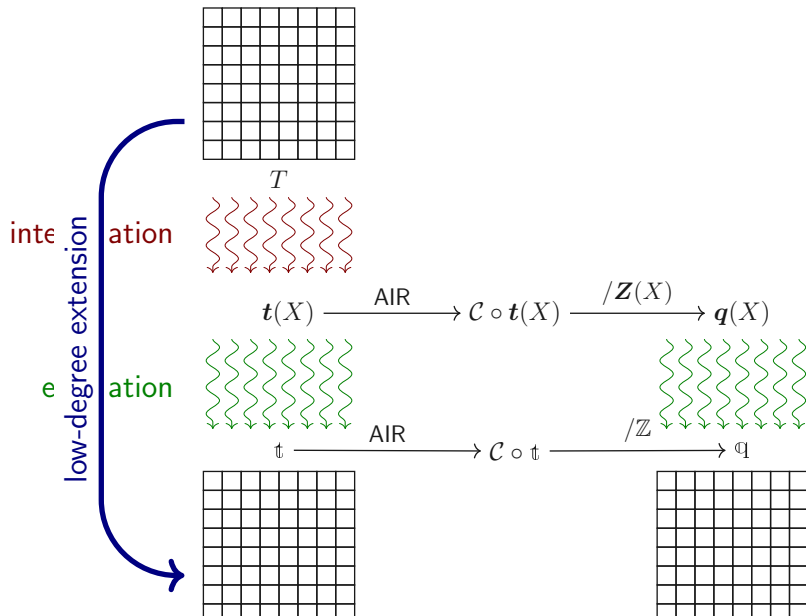


$$t(X) \xrightarrow{\text{AIR}} \mathcal{C} \circ t(X) \xrightarrow{/\mathbf{Z}(X)} \mathbf{q}(X)$$

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 $\mathbf{p} \in \mathbb{F}^{N/\rho}$

pointwise maps

low degree tests

simulate

Prover

$p(X)$

Verifier

$p(X)$

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

$y \leftarrow p(z)$

Prover

$\mathbf{p}$

Verifier

$\mathbf{p}$

Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

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

evaluation queries

Low-Degree IOP:

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

pointwise maps

low degree tests

simulate



Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

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

evaluation queries

Low-Degree IOP:

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

pointwise maps

low degree tests

simulate

Prover

$p(X)$

Verifier

$p(X)$

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

$y \leftarrow p(z)$

Prover

$\mathbf{p}$

Verifier

$\mathbf{p}$

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

$\xleftarrow{z}$
 $\xrightarrow{p(z)}$

$\mathbf{q} \leftarrow \text{DEEP}(\mathbf{p}, p(z), z)$

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

Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

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

evaluation queries

Low-Degree IOP:

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

pointwise maps

low degree tests

simulate

Prover

$p(X)$

Verifier

$p(X)$

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

$y \leftarrow p(z)$

Prover

$\mathbf{p}$

Verifier

$\mathbf{p}$

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

$\xleftarrow{z}$
 $\xrightarrow{p(z)}$

$\mathbf{q} \leftarrow \text{DEEP}(\mathbf{p}, p(z), z)$

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

$\mathbf{p}, \mathbf{q} \stackrel{?}{\in} \text{RS}$

Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

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

evaluation queries

Low-Degree IOP:

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

pointwise maps

low degree tests

simulate

Prover

$p(X)$

Verifier

$p(X)$

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

$y \leftarrow p(z)$

Prover

$\mathbf{p}$

Verifier

$\mathbf{p}$

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

$\xleftarrow{z}$
 $\xrightarrow{p(z)}$

$\mathbf{q} \leftarrow \text{DEEP}(\mathbf{p}, p(z), z)$

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

$\mathbf{p}, \mathbf{q} \stackrel{?}{\in} \text{RS}$

Soundness.

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

$$\therefore \varepsilon = 0.$$

Polynomial IOP to Low-Degree IOP (2)

Polynomial IOP:

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

evaluation queries

Low-Degree IOP:

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

pointwise maps

low degree tests

simulate

Prover

$p(X)$

Verifier

$p(X)$

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

$y \leftarrow p(z)$

Prover

$\mathbf{p}$

Verifier

$\mathbf{p}$

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

$\xleftarrow{z}$
 $\xrightarrow{p(z)}$

$\mathbf{q} \leftarrow \text{DEEP}(\mathbf{p}, p(z), z)$

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

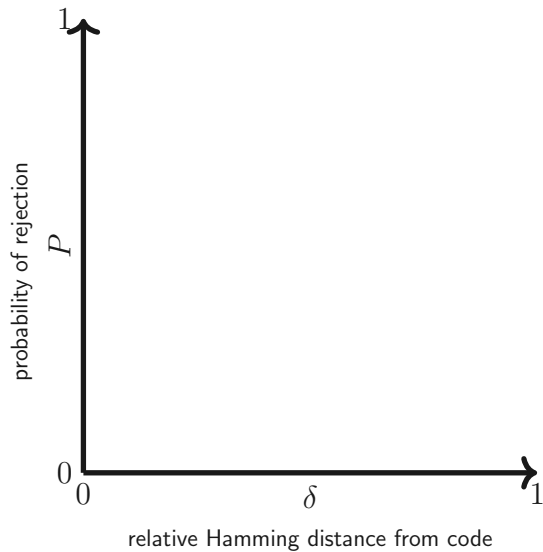
Soundness.

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

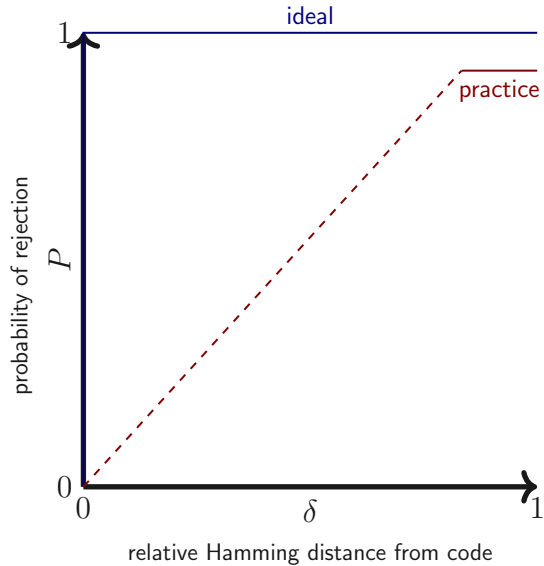
$\therefore \varepsilon = 0.$

$\mathbf{p}, \mathbf{q} \in \text{RS}$

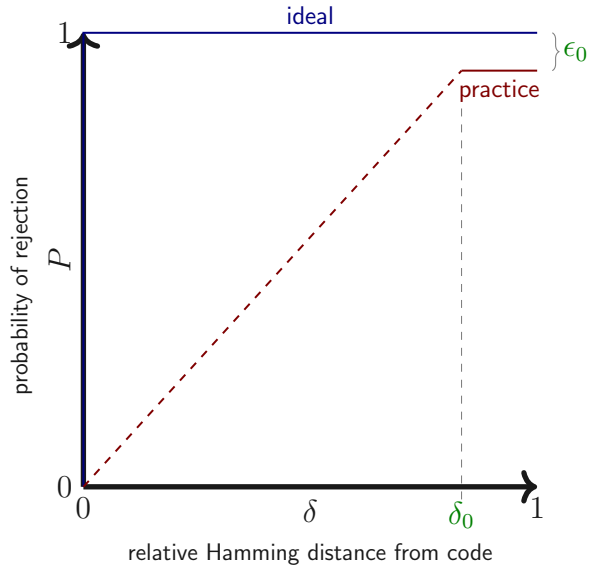
Distance Parameter



Distance Parameter



Distance Parameter



DEEP Soundness

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

DEEP Soundness

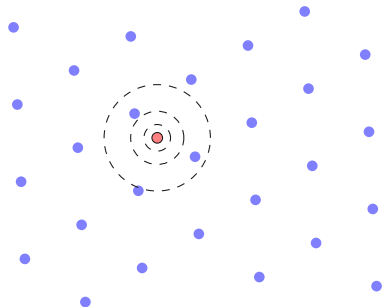
$$\Pr \left[\Delta \left(\left[\frac{\mathbf{p}[i]-y}{D[i]-z} \right]_i, \text{RS} \right) < \delta \mid \forall f(X) \in \text{list}(\mathbf{p}), f(z) \neq y \right] \leq \frac{(\#\text{list})^2}{2} \cdot \frac{\deg}{|\mathbb{F}| - N}$$

DEEP Soundness

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

$\#\text{list}$ depends on δ

Reed-Solomon List Decoding Conjecture



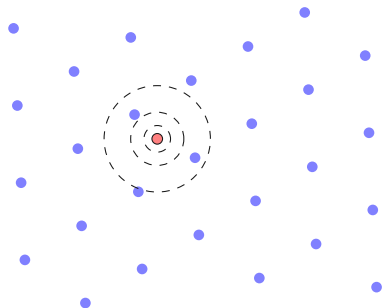
Reed-Solomon List Decoding Conjecture

decode an arbitrary point

for which radii δ

is the resulting list L

$|L| \leq \text{poly?}$



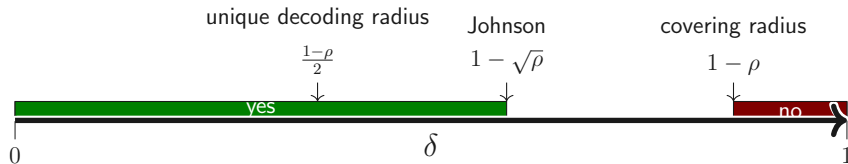
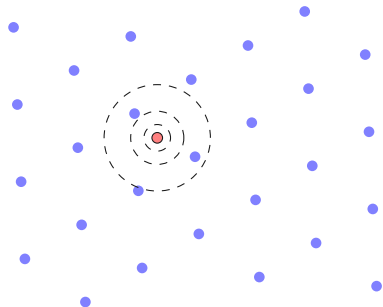
Reed-Solomon List Decoding Conjecture

decode an arbitrary point

for which radii δ

is the resulting list L

$|L| \leq \text{poly}$?



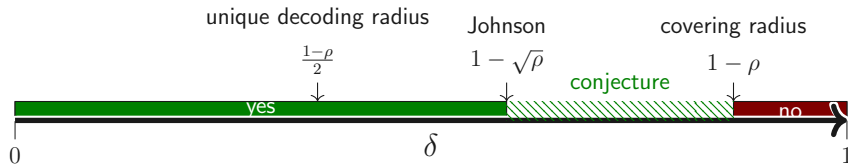
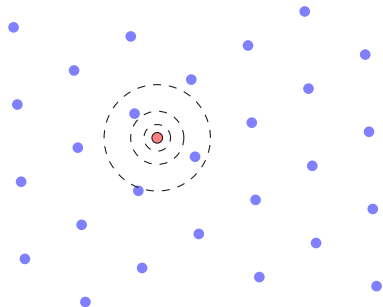
Reed-Solomon List Decoding Conjecture

decode an arbitrary point

for which radii δ

is the resulting list L

$|L| \leq \text{poly}$?



DEEP Soundness

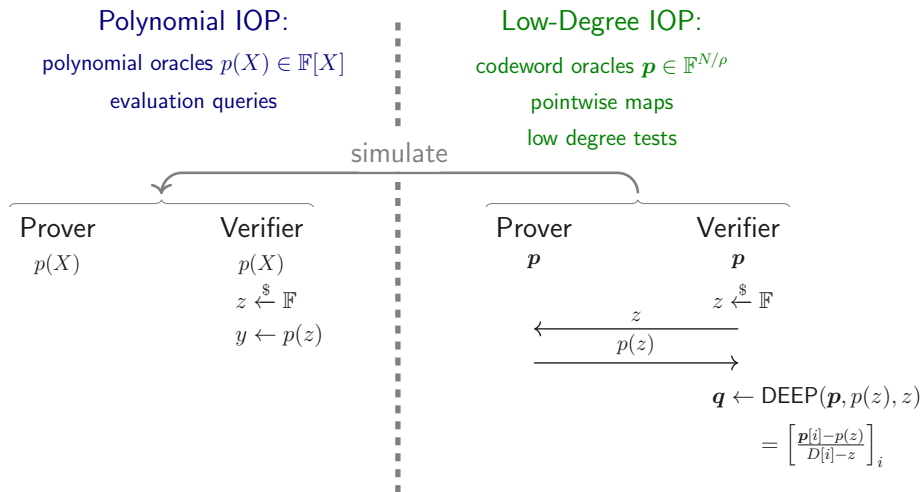
$$\Pr \left[\Delta \left(\left[\frac{\mathbf{p}[i]-y}{D[i]-z} \right]_i, \text{RS} \right) < \delta \mid \forall f(X) \in \text{list}(\mathbf{p}), f(z) \neq y \right] \leq \frac{(\#\text{list})^2}{2} \cdot \frac{\deg}{|\mathbb{F}| - N}$$

DEEP Soundness

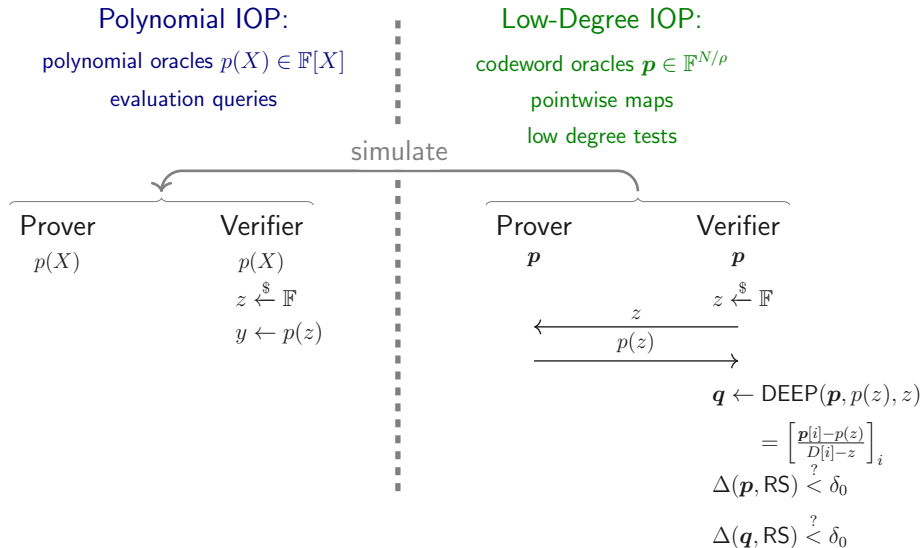
$$\Pr \left[\Delta \left(\left[\frac{\mathbf{p}[i]-y}{D[i]-z} \right]_i, \text{RS} \right) < \delta \mid \forall f(X) \in \text{list}(\mathbf{p}), f(z) \neq y \right] \leq \frac{(\#\text{list})^2}{2} \cdot \frac{\deg}{|\mathbb{F}| - N}$$

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

Polynomial IOP to Low-Degree IOP (3)



Polynomial IOP to Low-Degree IOP (3)



Polynomial IOP to Low-Degree IOP (3)

Polynomial IOP:

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

evaluation queries

Low-Degree IOP:

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

pointwise maps

low degree tests

simulate

Prover

$p(X)$

Verifier

$p(X)$

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

$y \leftarrow p(z)$

Prover

$\mathbf{p}$

Verifier

$\mathbf{p}$

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

$\xleftarrow{z}$
 $\xrightarrow{p(z)}$

Soundness.

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

$\mathbf{q} \leftarrow \text{DEEP}(\mathbf{p}, p(z), z)$

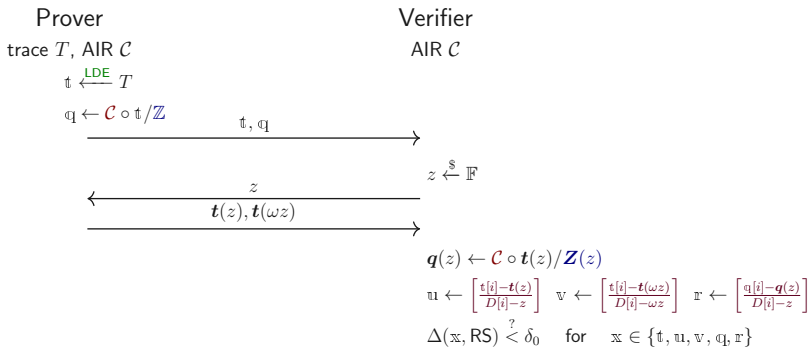
$= \left[\frac{\mathbf{p}[i] - p(z)}{D[i] - z} \right]_i$

$\Delta(\mathbf{p}, \text{RS}) \stackrel{?}{<} \delta_0$

$\Delta(\mathbf{q}, \text{RS}) \stackrel{?}{<} \delta_0$

DEEP-ALI + DEEP

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



DEEP-ALI + DEEP

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

Soundness.

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

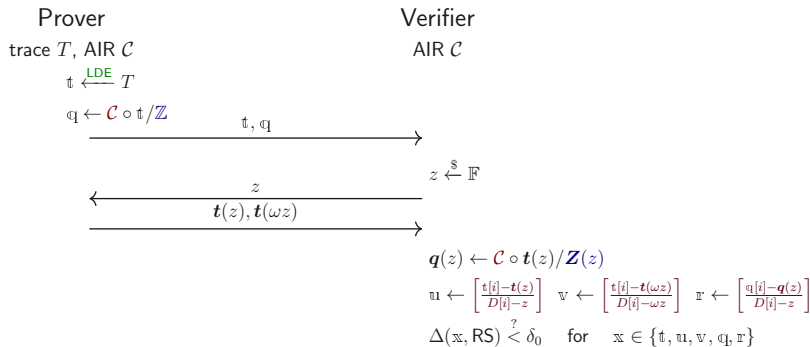


Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

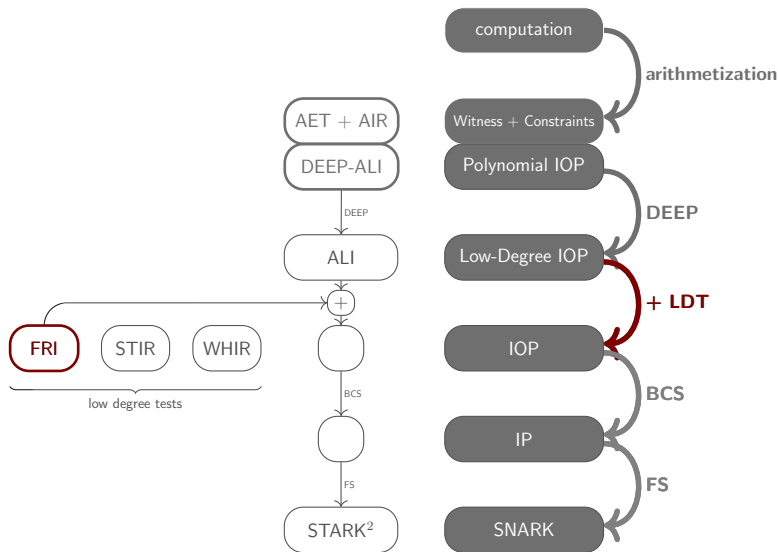
Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

STARK Compilation Pipeline (Low-Degree Testing)



Intermezzo: Batching

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

Intermezzo: Batching

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

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

Intermezzo: Batching

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

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

Intermezzo: Batching

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

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

Intermezzo: Batching

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

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

Intermezzo: Batching

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

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

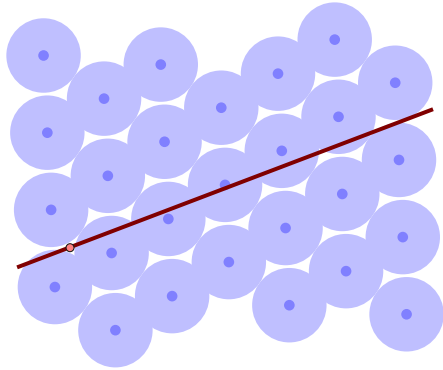
Intermezzo: Batching

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

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

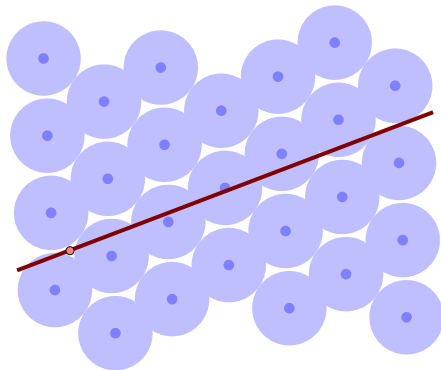
$$\Pr_r [\Delta((1-r) \cdot \mathbf{u} + r \cdot \mathbf{v}, \text{RS}) < \delta \mid \Delta(\mathbf{u}, \text{RS}) > \delta \vee \Delta(\mathbf{v}, \text{RS}) > \delta]$$

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



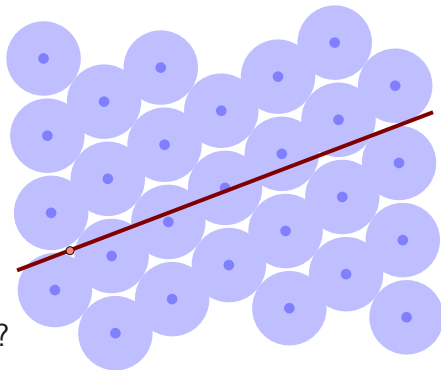
Reed-Solomon Proximity Gap

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?

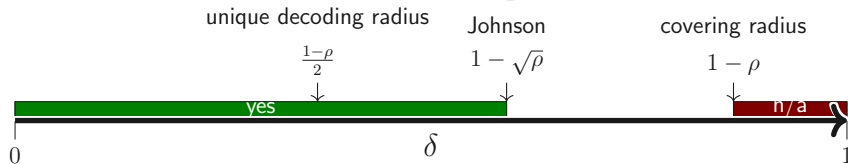
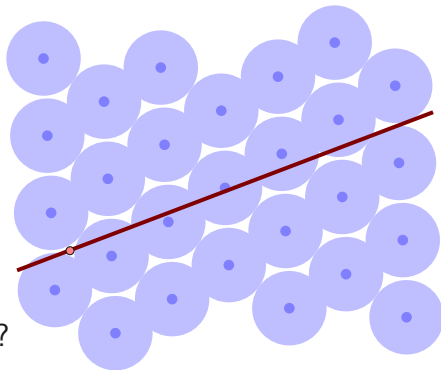


Reed-Solomon Proximity Gap

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?

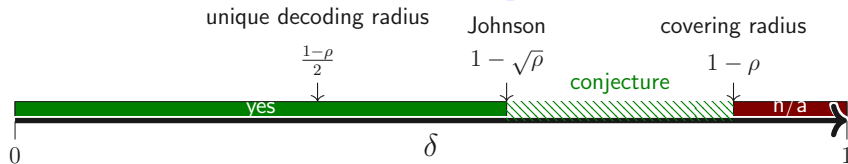
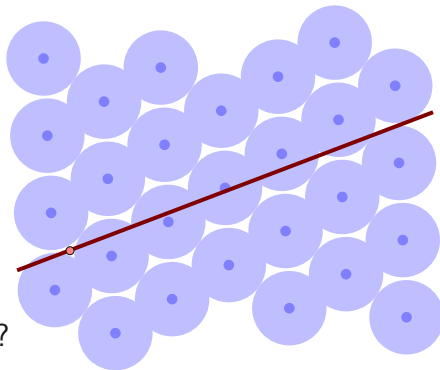


Reed-Solomon Proximity Gap

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

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

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

Intermezzo: Batching

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

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

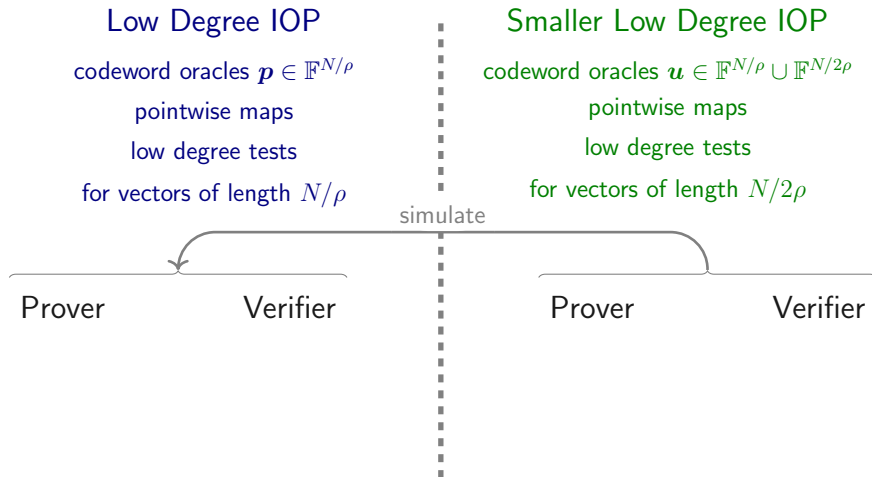
$\exists(\epsilon_0, \delta_0)$ -gap

$\Leftrightarrow$

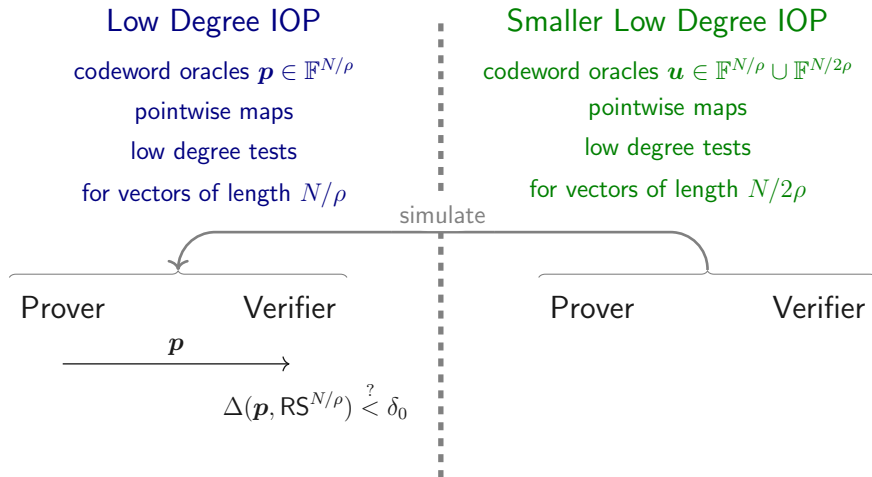
$$\Pr_r [\Delta((1-r) \cdot \mathbf{u} + r \cdot \mathbf{v}, \text{RS}) < \delta_0 \mid \Delta(\mathbf{u}, \text{RS}) > \delta_0 \vee \Delta(\mathbf{v}, \text{RS}) > \delta_0] \leq \epsilon_0$$

</Intermezzo>

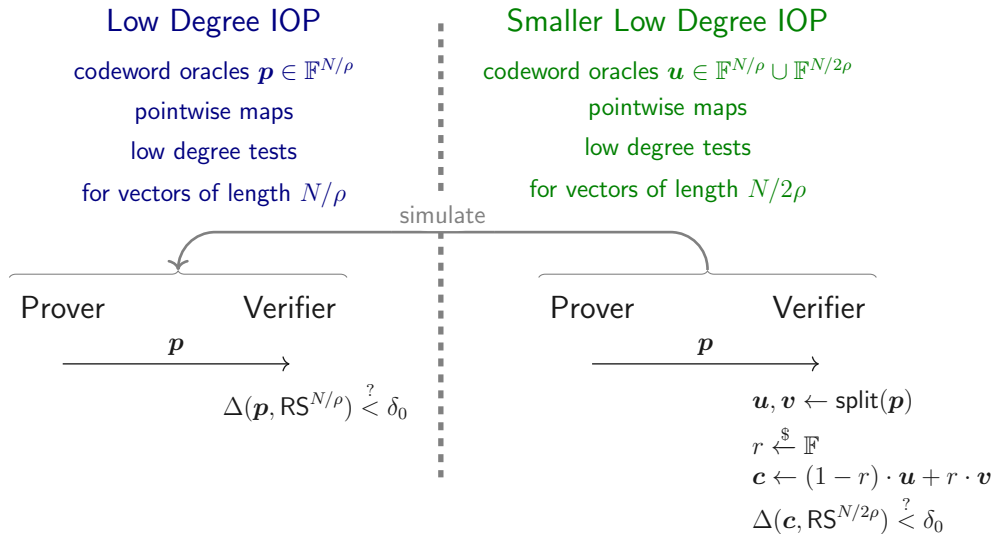
Low Degree IOP to Smaller Low Degree IOP



Low Degree IOP to Smaller Low Degree IOP



Low Degree IOP to Smaller Low Degree IOP



Low Degree IOP to Smaller Low Degree IOP

Low Degree IOP

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

pointwise maps

low degree tests

for vectors of length N/ρ

Smaller Low Degree IOP

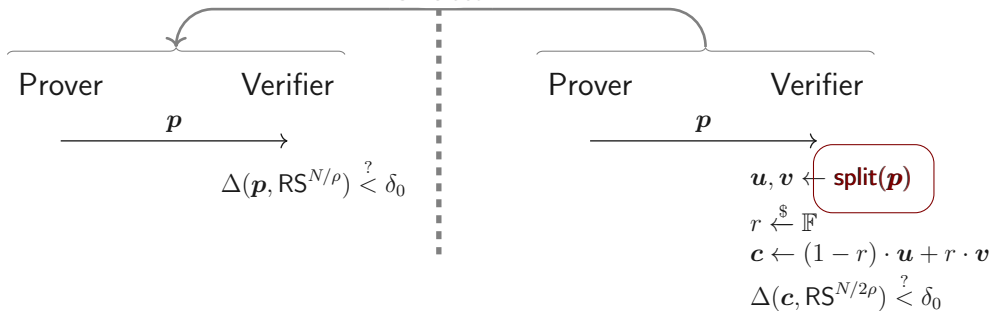
codeword oracles $u \in \mathbb{F}^{N/\rho} \cup \mathbb{F}^{N/2\rho}$

pointwise maps

low degree tests

for vectors of length $N/2\rho$

simulate



Split

→ half the length; same rate

$$\text{split} : \mathbb{R}S^{N/\rho} \rightarrow \mathbb{R}S^{N/2\rho} \times \mathbb{R}S^{N/2\rho}$$

Split

→ half the length; same rate

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

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

Split

→ half the length; same rate

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

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

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

Split

→ half the length; same rate

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

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

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

Soundness.

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

Split

→ half the length; same rate

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

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

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

Soundness.

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

- no probability variables $\Rightarrow \Pr \in \{0, 1\}$
- local map is linear over $\mathbb{F}^2$ and invertible, so $0 \leftrightarrow 0$

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

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

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

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

Problem: verifier work explodes exponentially

split^{log N}

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

Problem: verifier work explodes exponentially

Solution: prover helps

split^{log N}

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

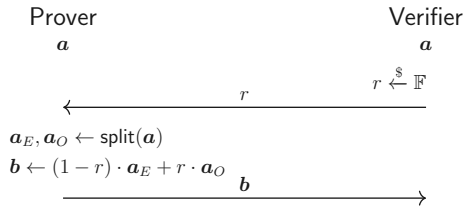
Problem: verifier work explodes exponentially

Solution: prover helps

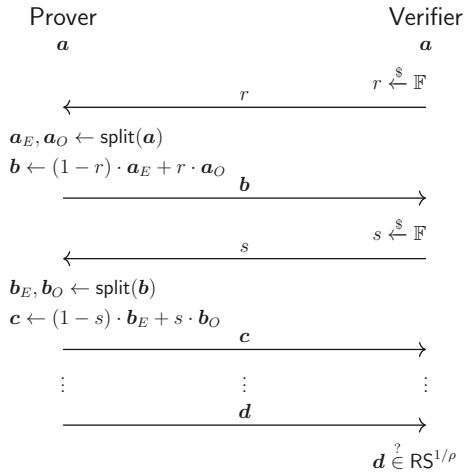
Problem: malicious help

Solution: verifier checks help

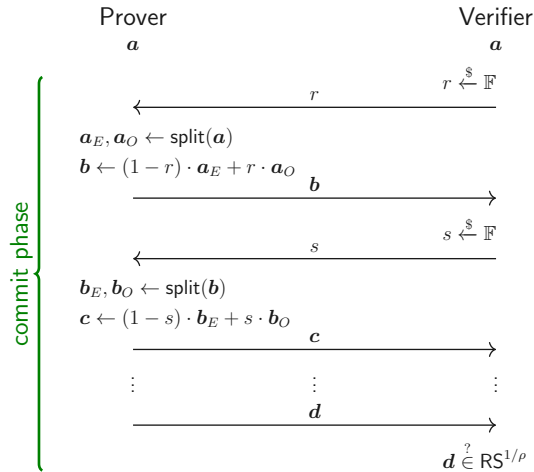
FRI: Fast Reed-Solomon IOPP



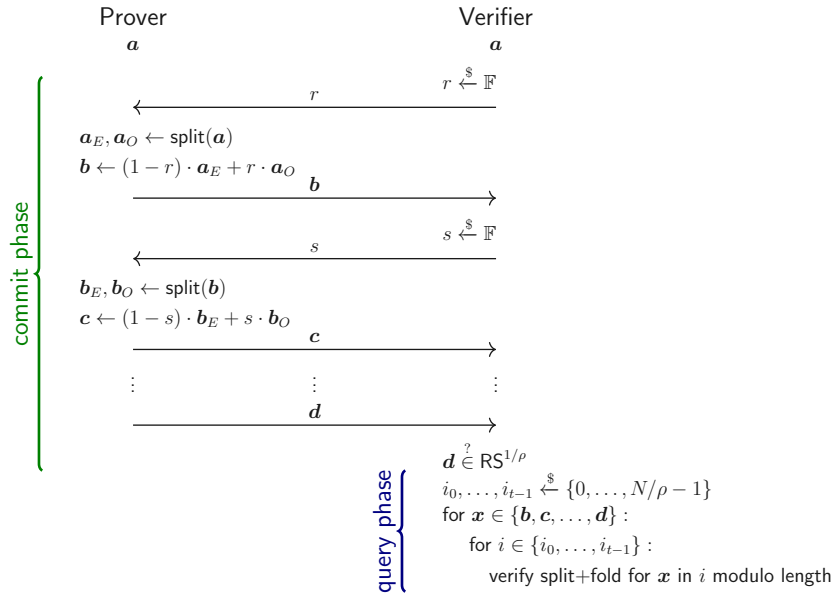
FRI: Fast Reed-Solomon IOPP



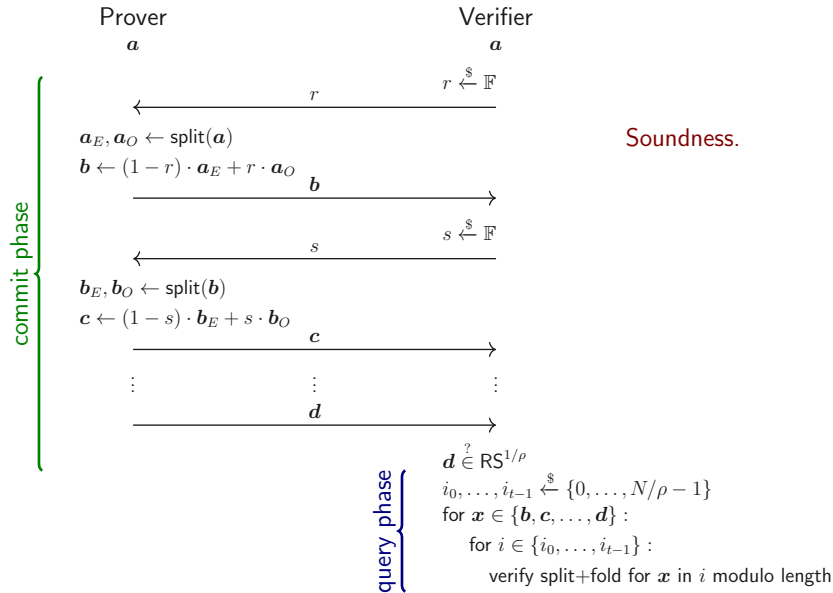
FRI: Fast Reed-Solomon IOPP



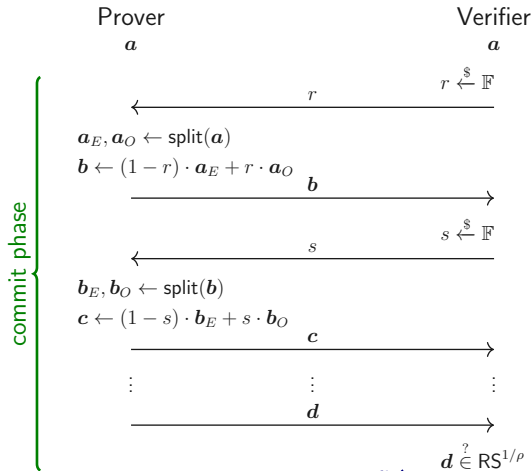
FRI: Fast Reed-Solomon IOPP



FRI: Fast Reed-Solomon IOPP



FRI: Fast Reed-Solomon IOPP



Soundness.

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

query phase

$$d \stackrel{?}{\in} \text{RS}^{1/\rho}$$

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

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

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

verify split+fold for x in i modulo length

FRI: Visual Intuition



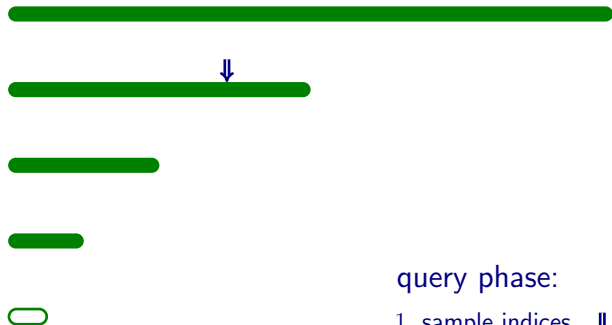
FRI: Visual Intuition



commit phase:

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

FRI: Visual Intuition



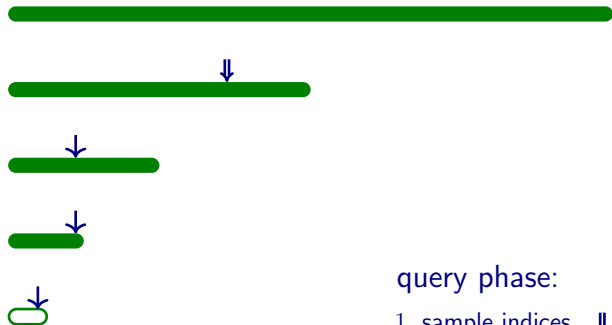
query phase:

1. sample indices ↓

commit phase:

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

FRI: Visual Intuition



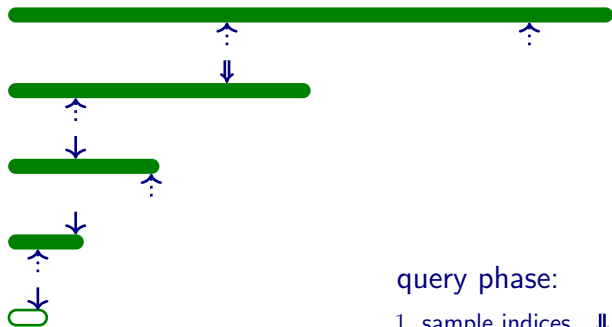
commit phase:

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

query phase:

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

FRI: Visual Intuition



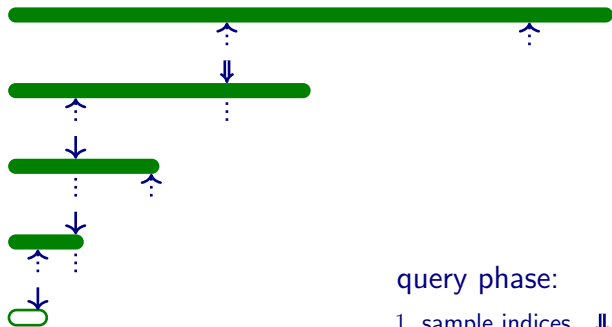
commit phase:

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 $\Uparrow$

FRI: Visual Intuition



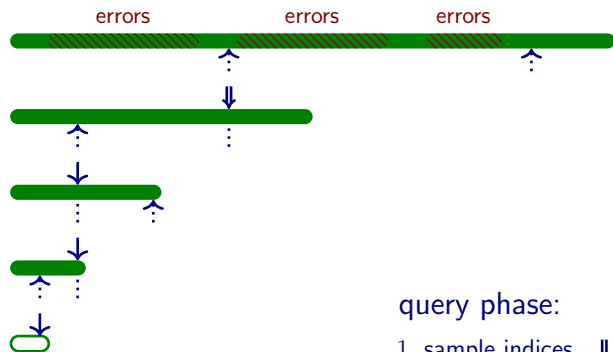
commit phase:

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 $\Uparrow$
3. check split+fold $\Downarrow \Downarrow$
4. reuse result $\vdots$

FRI: Visual Intuition



Soundness:

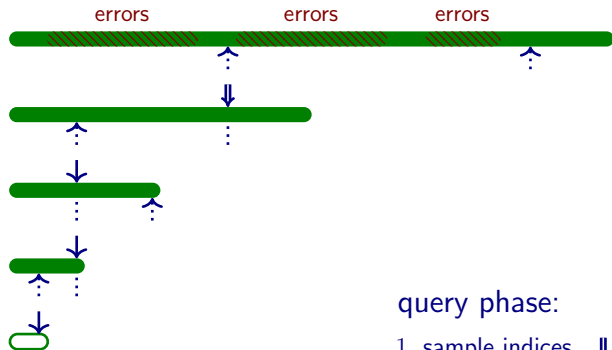
query phase:

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

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:

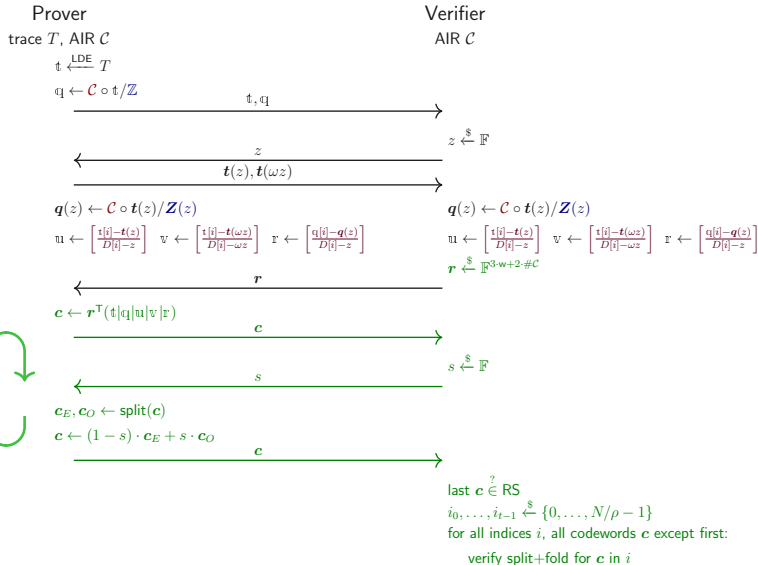
1. sample indices $\Downarrow$
2. indices cascade modulo length $\Downarrow$
3. query codewords $\Uparrow$
3. check split+fold $\Downarrow \Downarrow$
4. reuse result $\vdots$

Soundness:

probability that:

- 1 index points to error: $\geq \delta$
- 1 index points to non-error: $\leq 1 - \delta$
- all t indices point to non-errors: $\leq (1 - \delta)^t$

DEEP-ALI + DEEP + FRI



DEEP-ALI + DEEP + FRI

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

$$t \xleftarrow{\text{LDE}} T$$

$$q_1 \leftarrow \mathcal{C} \circ t / \mathcal{Z}$$

$$t, q_1$$

Verifier
AIR $\mathcal{C}$

Soundness.

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

Schwartz-Zippel
list-decoding

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

$$t(z), t(\omega z)$$

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

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

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

$$u_1 \leftarrow \left[\frac{t[i] - t(z)}{D[i] - z} \right] \quad v \leftarrow \left[\frac{t[i] - t(\omega z)}{D[i] - \omega z} \right] \quad r \leftarrow \left[\frac{q[i] - q(z)}{D[i] - z} \right]$$

$$u_1 \leftarrow \left[\frac{t[i] - t(z)}{D[i] - z} \right] \quad v \leftarrow \left[\frac{t[i] - t(\omega z)}{D[i] - \omega z} \right] \quad r \leftarrow \left[\frac{q[i] - q(z)}{D[i] - z} \right]$$

$$r \xleftarrow{\$} \mathbb{F}^{3 \cdot w + 2 \cdot \#C}$$

$$r$$

$$c \leftarrow r^T (t|q_1|u_1|v|r)$$

$$c$$

$$s \xleftarrow{\$} \mathbb{F}$$

$$s$$

$$c_E, c_O \leftarrow \text{split}(c)$$

$$c \leftarrow (1-s) \cdot c_E + s \cdot c_O$$

$$c$$

last $c \in \text{RS}$

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

for all indices i , all codewords c except first:

verify split+fold for c in i

DEEP-ALI + DEEP + FRI

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

$$t \xleftarrow{\text{LDE}} T$$

$$q_1 \leftarrow \mathcal{C} \circ t / \mathbb{Z}$$

$$t, q_1$$

Verifier
AIR $\mathcal{C}$

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

$$z$$

$$t(z), t(\omega z)$$

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

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

$$u_1 \leftarrow \left[\frac{t[i] - t(z)}{D[i] - z} \right] \quad v \leftarrow \left[\frac{t[i] - t(\omega z)}{D[i] - \omega z} \right] \quad r \leftarrow \left[\frac{q[i] - q(z)}{D[i] - z} \right]$$

$$u_1 \leftarrow \left[\frac{t[i] - t(z)}{D[i] - z} \right] \quad v \leftarrow \left[\frac{t[i] - t(\omega z)}{D[i] - \omega z} \right] \quad r \leftarrow \left[\frac{q[i] - q(z)}{D[i] - z} \right]$$

$$r \xleftarrow{\$} \mathbb{F}^{3 \cdot w + 2 \cdot \# \mathcal{C}}$$

$$r$$

$$c \leftarrow r^T (t | q_1 | u_1 | v | r)$$

$$c$$

$$s \xleftarrow{\$} \mathbb{F}$$

$$s$$

$$c_E, c_O \leftarrow \text{split}(c)$$

$$c \leftarrow (1 - s) \cdot c_E + s \cdot c_O$$

$$c$$

Soundness.

$$\frac{N-1}{|\mathbb{F}|}$$

$$+ \epsilon_{\text{DEEP}}(\delta_0)$$

$$+ (3 \cdot w + 2 \cdot \# \mathcal{C}) \cdot \epsilon_{\text{GAP}}(\delta_0)$$

$$+ r \cdot \epsilon_{\text{GAP}}(\delta_0)$$

$$+ (1 - \delta_0)^t$$

Schwartz-Zippel

list-decoding

proximity gap for batch

proximity gap for folding

bad indices



last $c \in ?$ RS

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

for all indices i , all codewords c except first:

verify split+fold for c in i

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

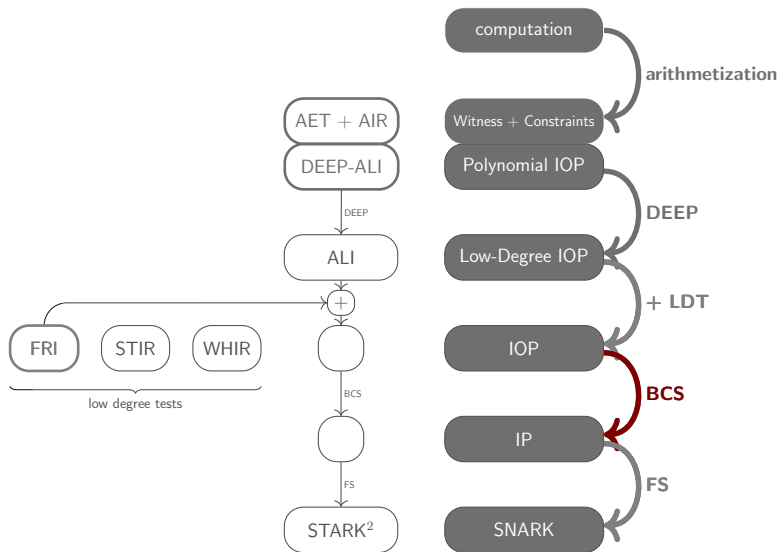
Low Degree Testing

BCS Transform

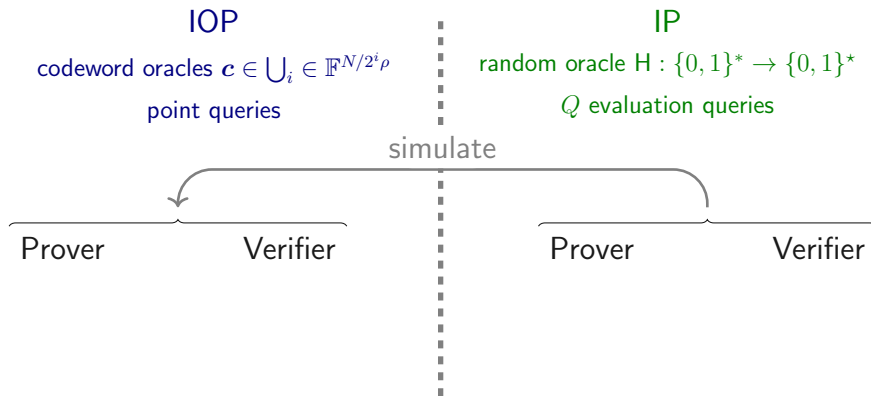
Fiat-Shamir Transform

Preview

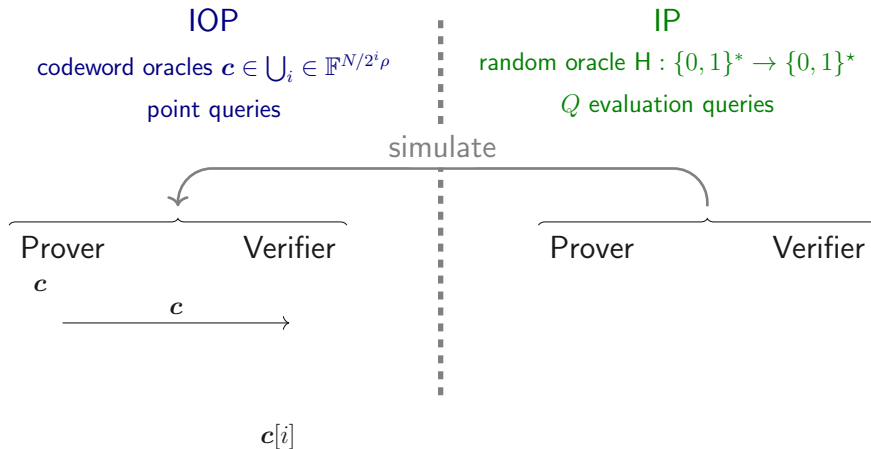
STARK Compilation Pipeline (BCS Transform)



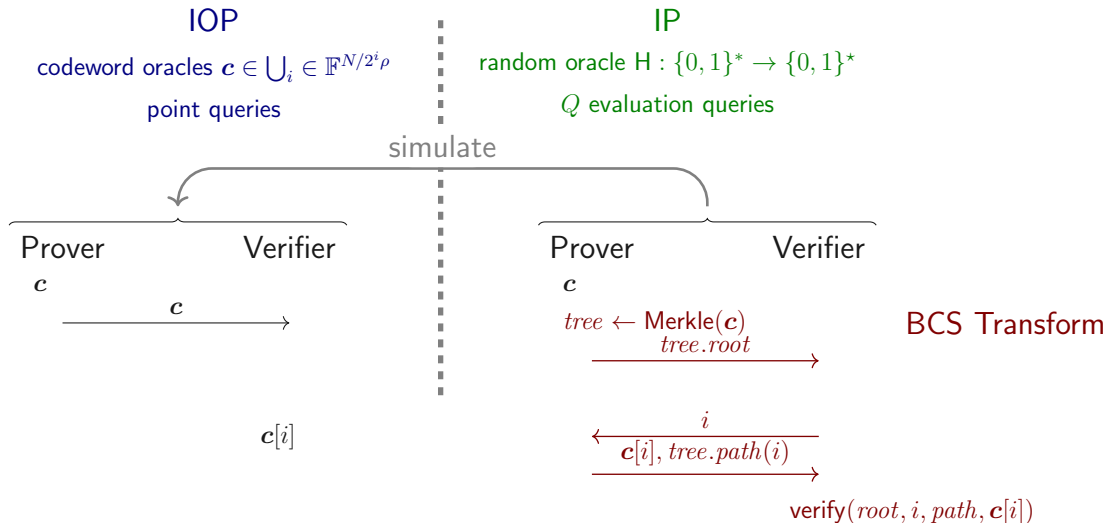
IOP to IP



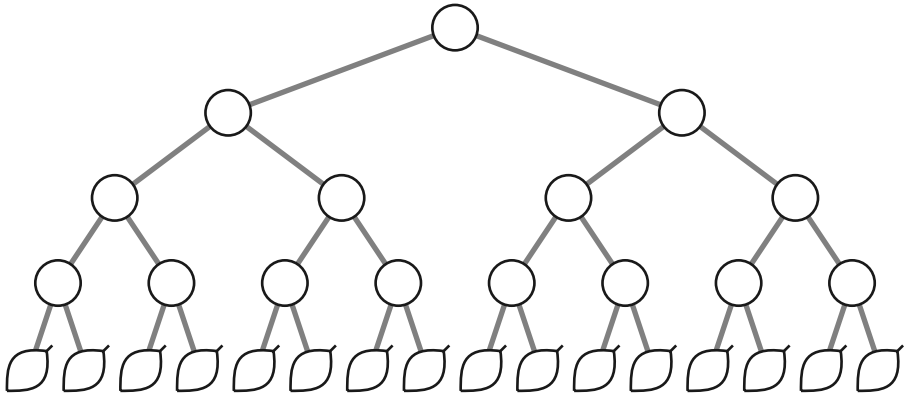
IOP to IP



IOP to IP

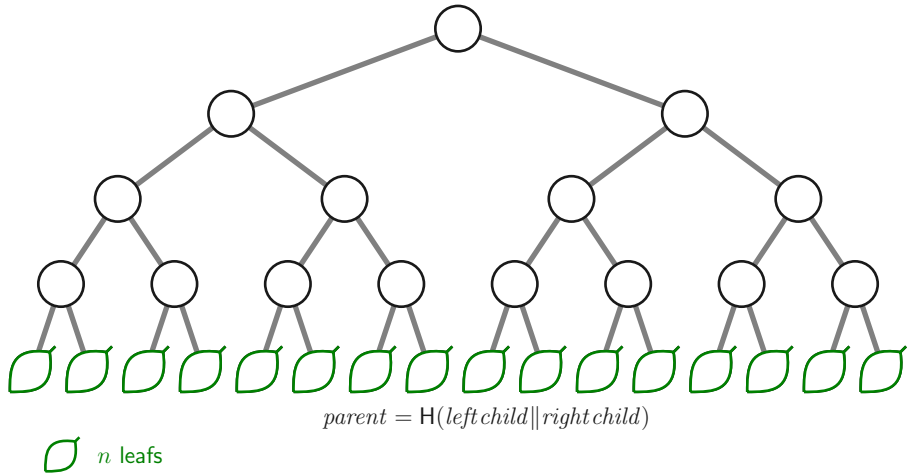


Merkle Tree

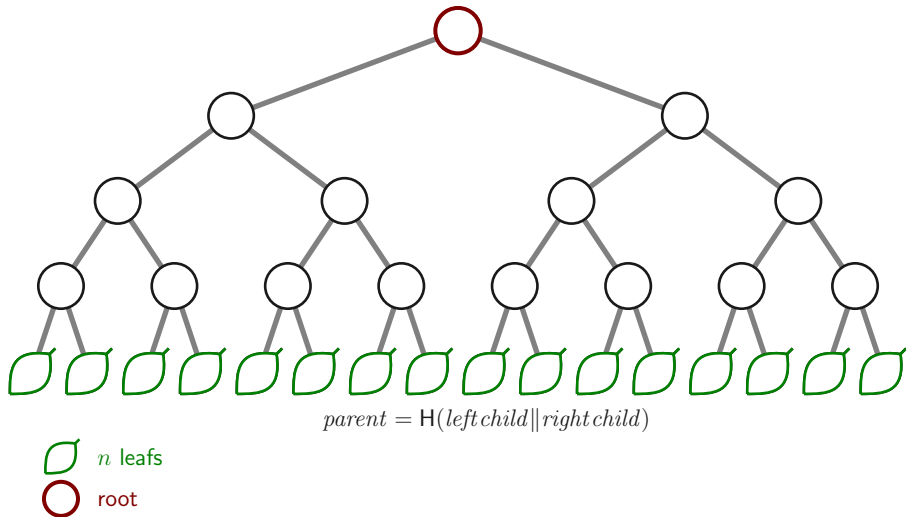


$$\text{parent} = H(\text{left child} \parallel \text{right child})$$

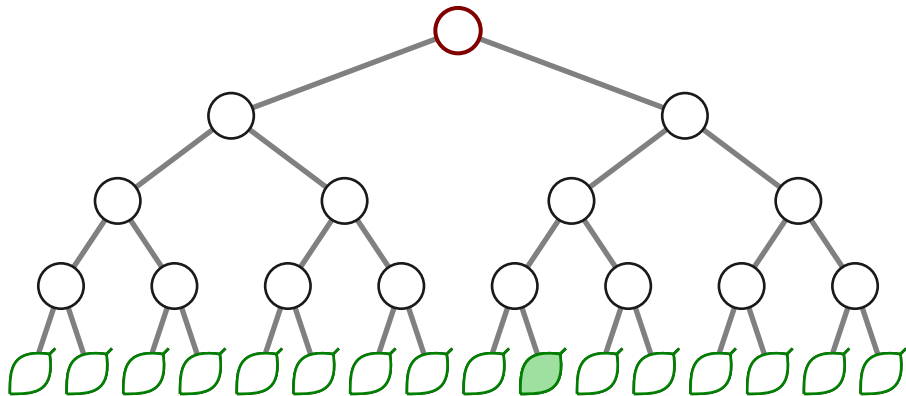
Merkle Tree



Merkle Tree



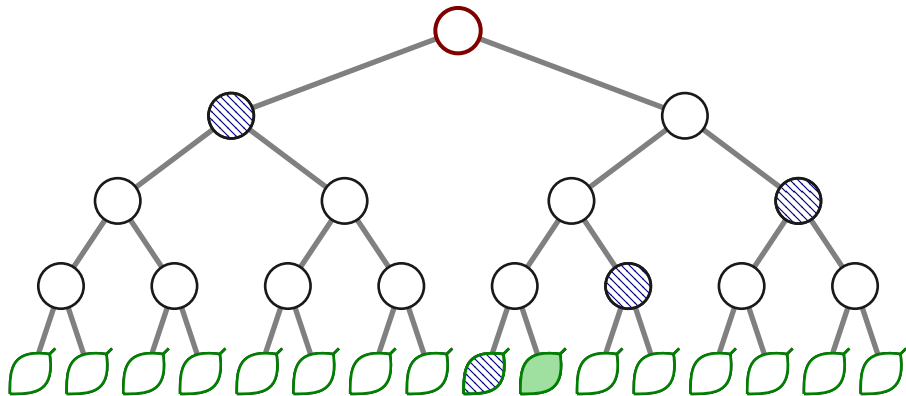
Merkle Tree



$$\text{parent} = H(\text{left child} \parallel \text{right child})$$

-  n leafs
-  root
-  opened leaf

Merkle Tree



$$\text{parent} = H(\text{left child} \parallel \text{right child})$$

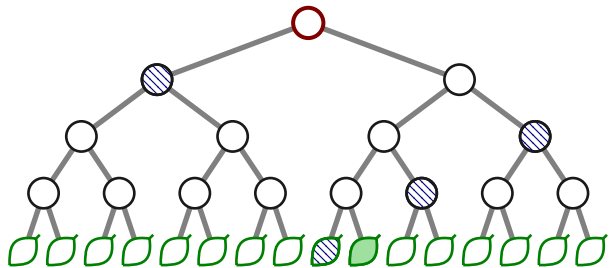
 n leafs

 root

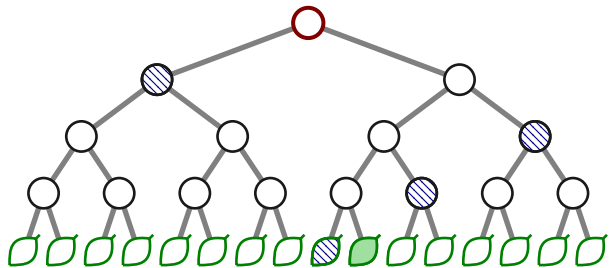
 opened leaf

 authentication path length $\log n$

Merkle Tree Commitment Scheme

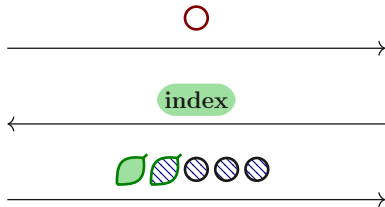


Merkle Tree Commitment Scheme

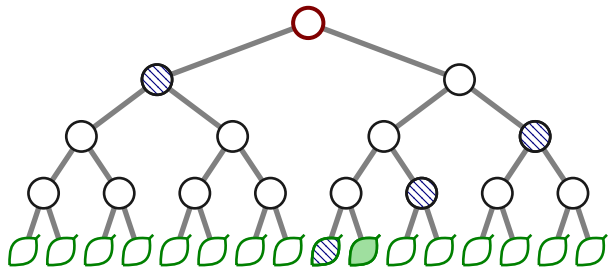


Prover

Verifier



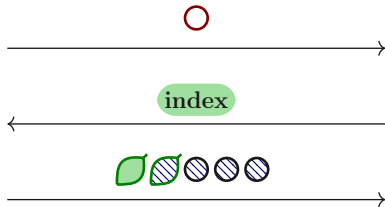
Merkle Tree Commitment Scheme



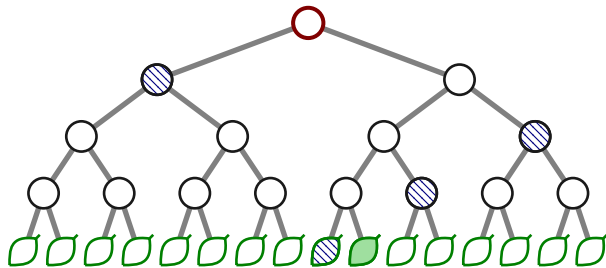
Soundness.

Prover

Verifier

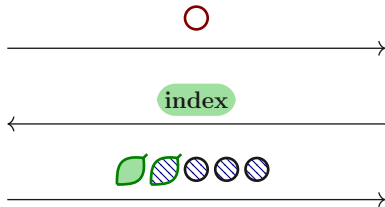


Merkle Tree Commitment Scheme



Prover

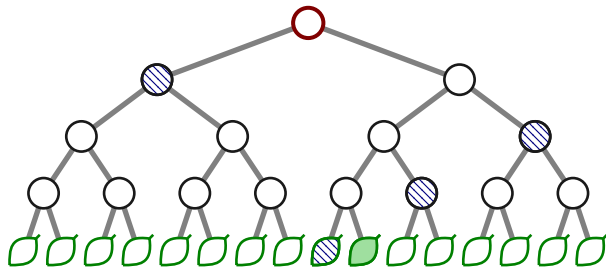
Verifier



Soundness.

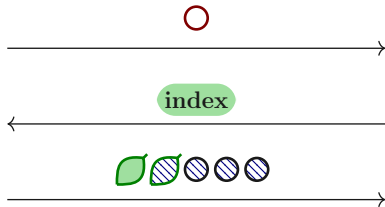
$\Pr[\text{Verifier} \checkmark \mid \text{bad leaf}]$

Merkle Tree Commitment Scheme



Prover

Verifier

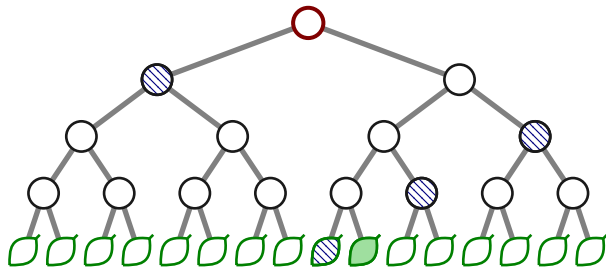


Soundness.

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

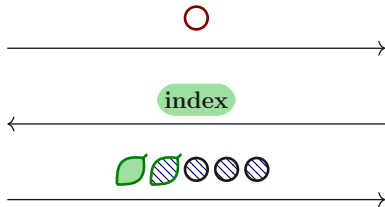
$\leq \Pr[\text{hash collision}]$

Merkle Tree Commitment Scheme



Prover

Verifier



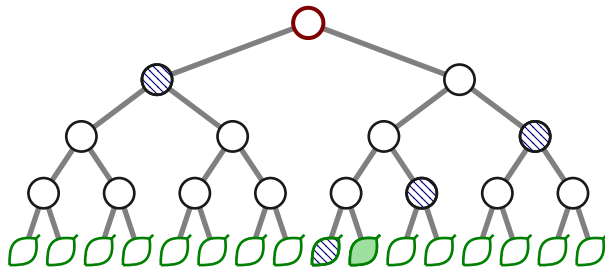
Soundness.

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

$\leq \Pr[\text{hash collision}]$

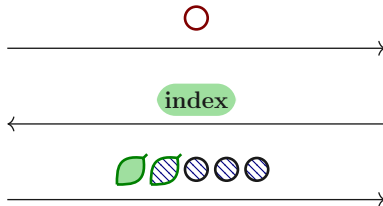
$$= \frac{Q \cdot (Q-1)}{2^\lambda}$$

Merkle Tree Commitment Scheme



Prover

Verifier



Soundness.

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

$\leq \Pr[\text{hash collision}]$

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

Q : # hash evaluations

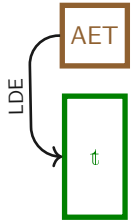
λ : # bits in output

STARK Diagram



algebraic execution trace

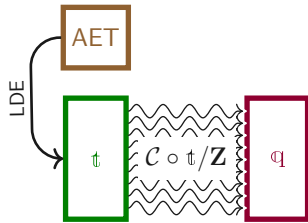
STARK Diagram



low-degree extension

low-degree extended trace

STARK Diagram

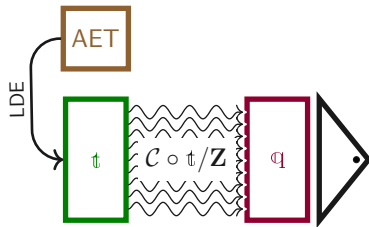


composition with AIR constraints

division by zeroifiers

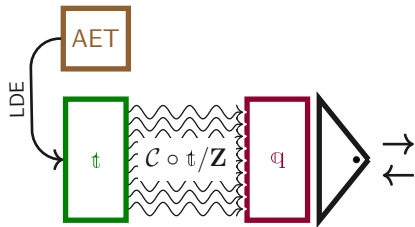
quotients

STARK Diagram



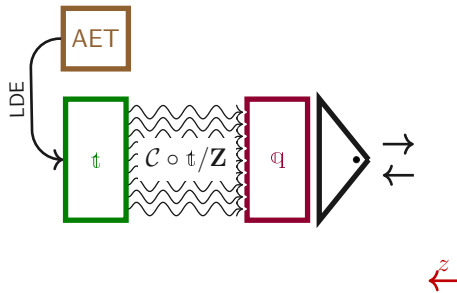
build Merkle tree

STARK Diagram



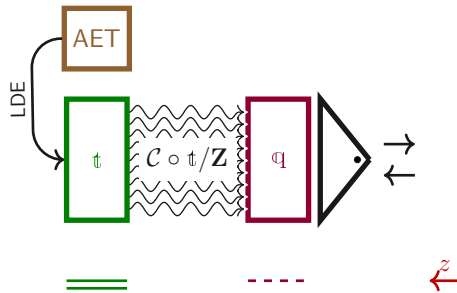
interact with verifier

STARK Diagram



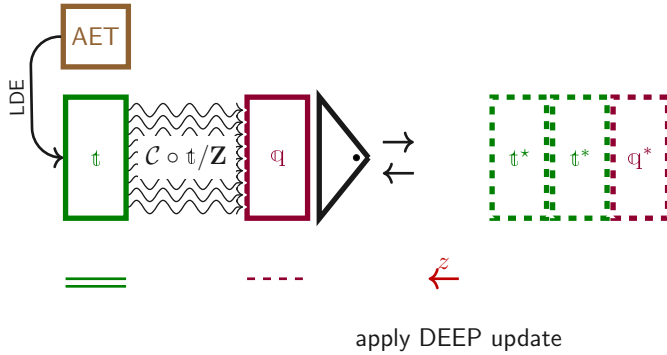
sample out-of-domain point

STARK Diagram

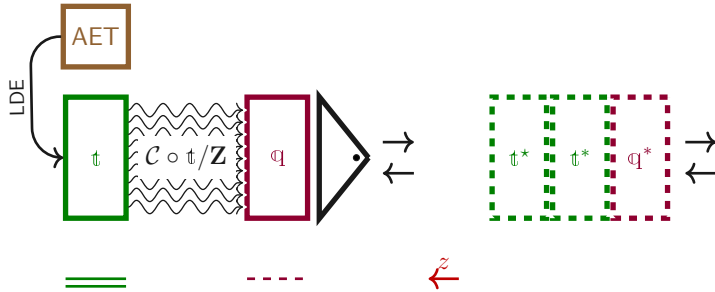


produce out-of-domain rows

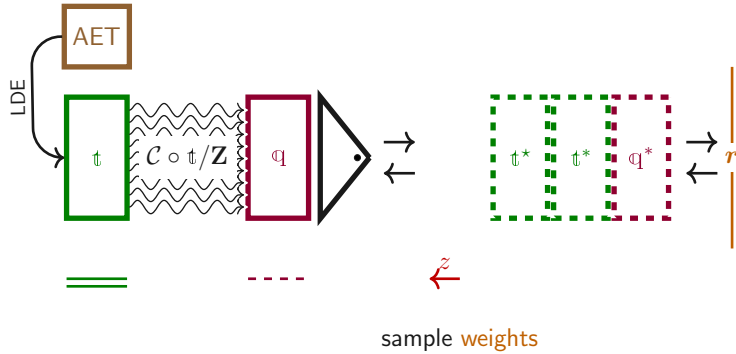
STARK Diagram



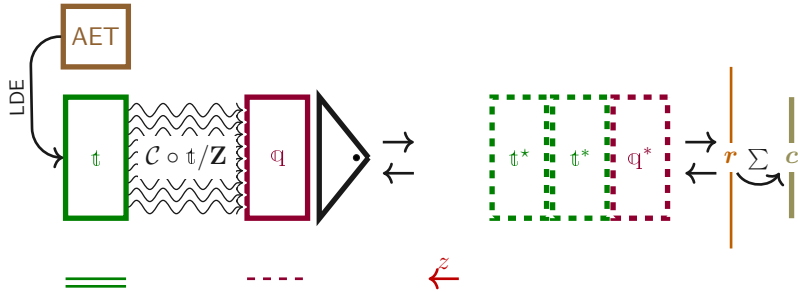
STARK Diagram



STARK Diagram

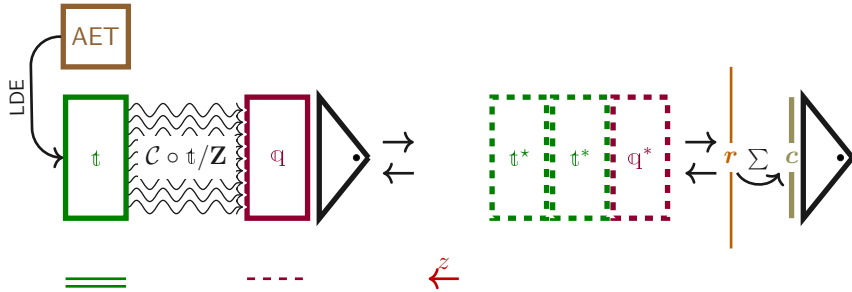


STARK Diagram



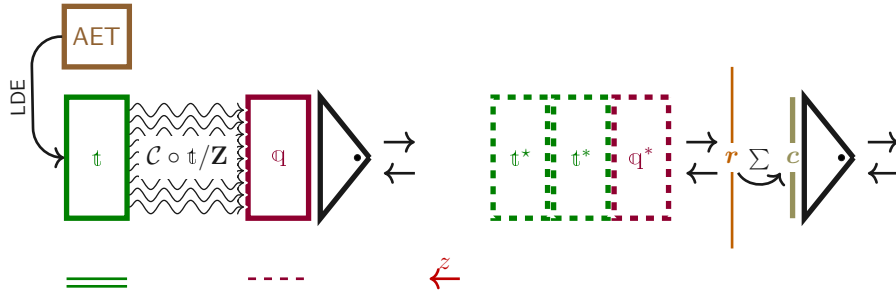
random linear combination

STARK Diagram



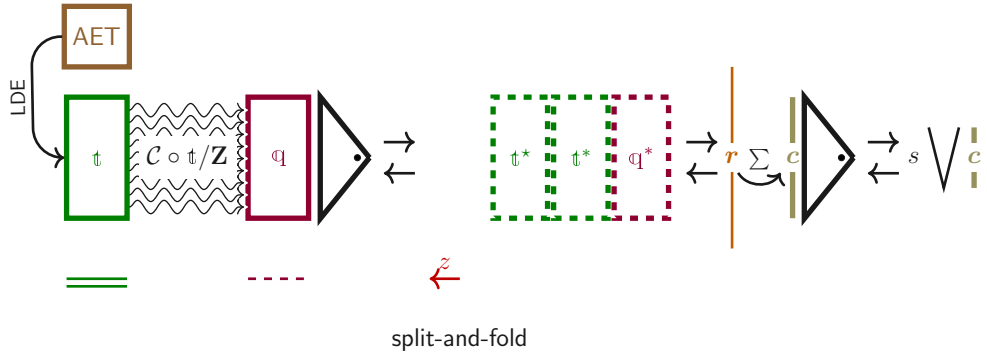
build Merkle tree

STARK Diagram

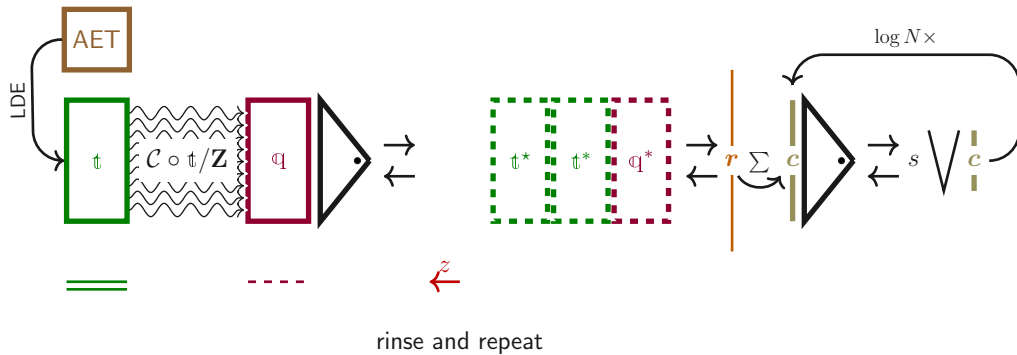


interact with verifier

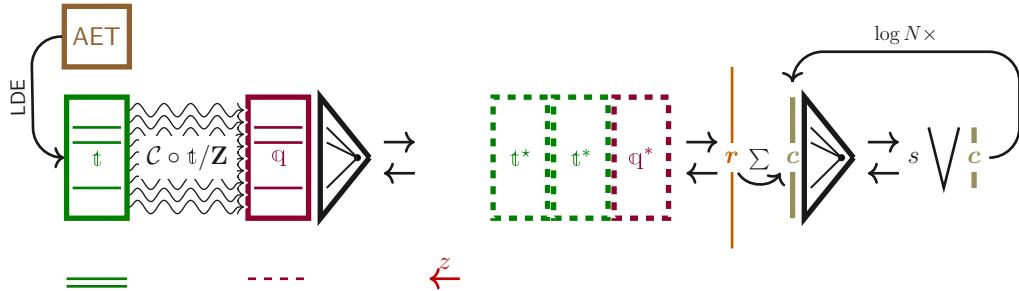
STARK Diagram



STARK Diagram

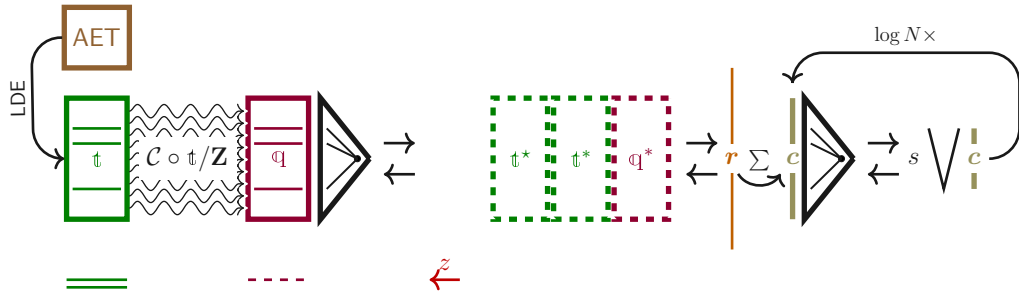


STARK Diagram

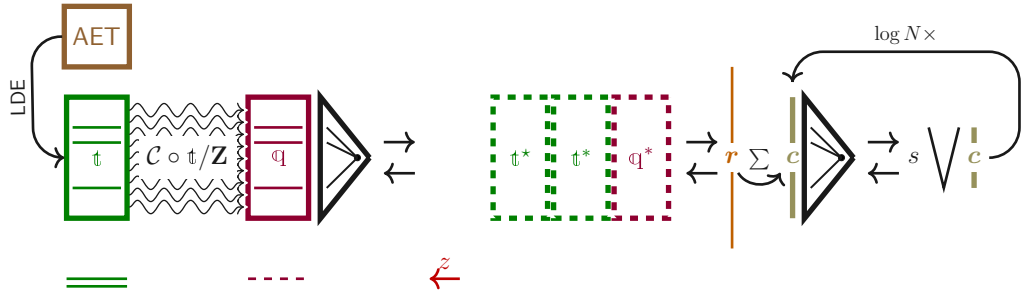


obtain FRI indices
open indicated rows

STARK Diagram

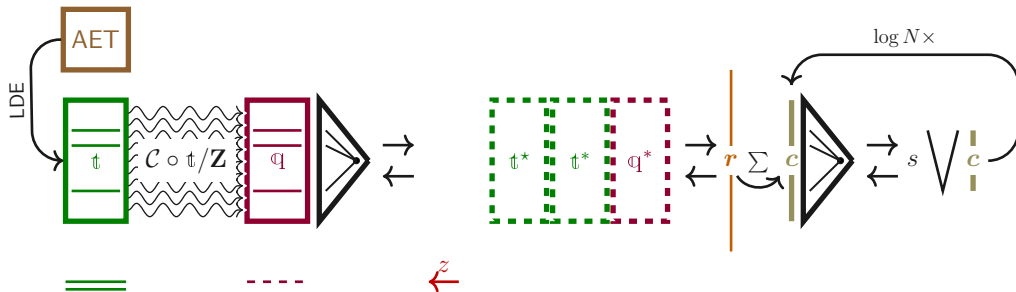


STARK Diagram



Soundness.

STARK Diagram



Soundness.

$$\text{DEEP-ALI} \left\{ \frac{N-1}{|\mathbb{F}|} + \epsilon_{\text{DEEP}}(\delta_0) \right.$$

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

$$\text{BCS} \left\{ +\frac{Q \cdot (Q-1)}{2^\lambda} \right.$$

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

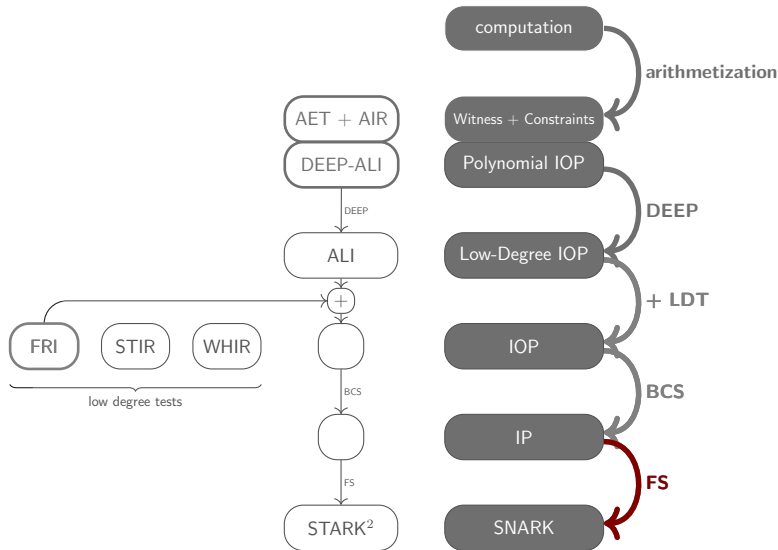
Low Degree Testing

BCS Transform

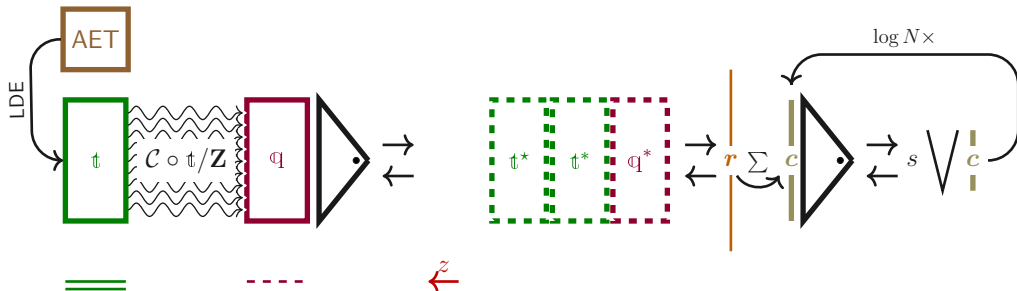
Fiat-Shamir Transform

Preview

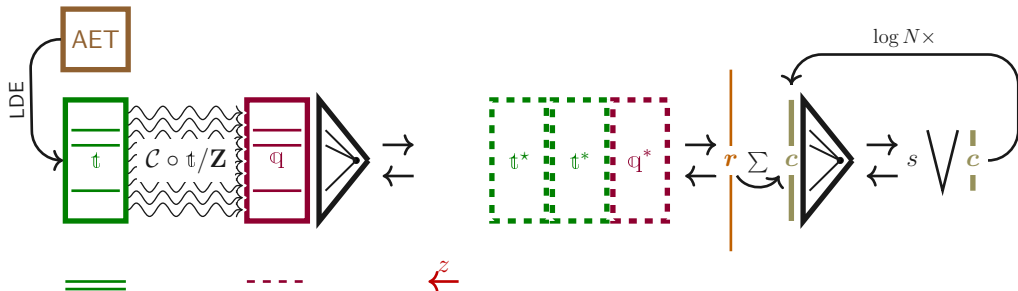
STARK Compilation Pipeline (Fiat-Shamir Transform)



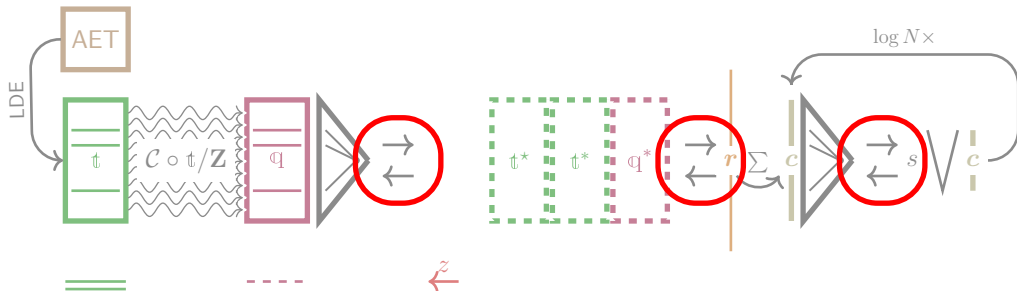
(Non-)Interactive STARK



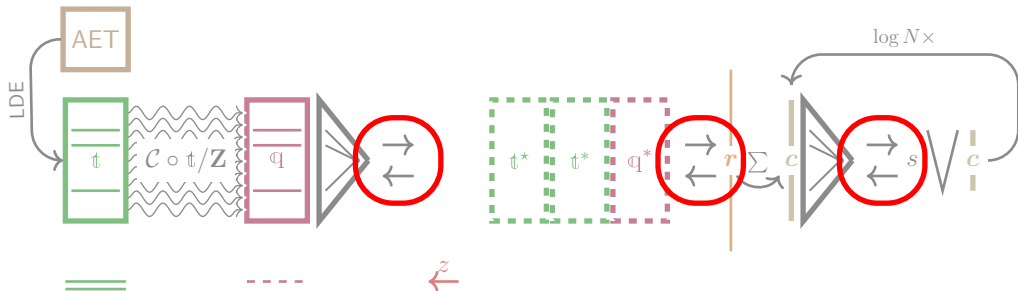
(Non-)Interactive STARK



(Non-)Interactive STARK



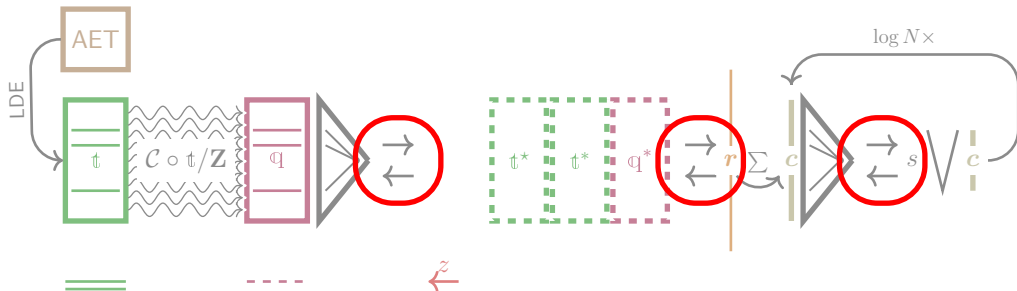
(Non-)Interactive STARK



Interactivity is a big problem in practice.

Can remove?

(Non-)Interactive STARK

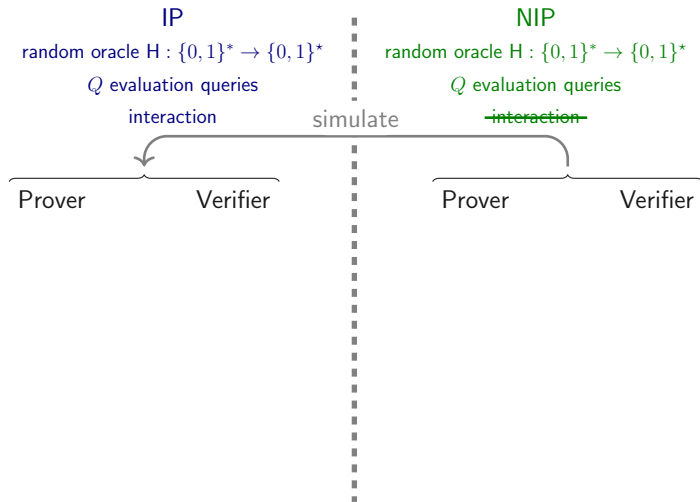


Interactivity is a big problem in practice.

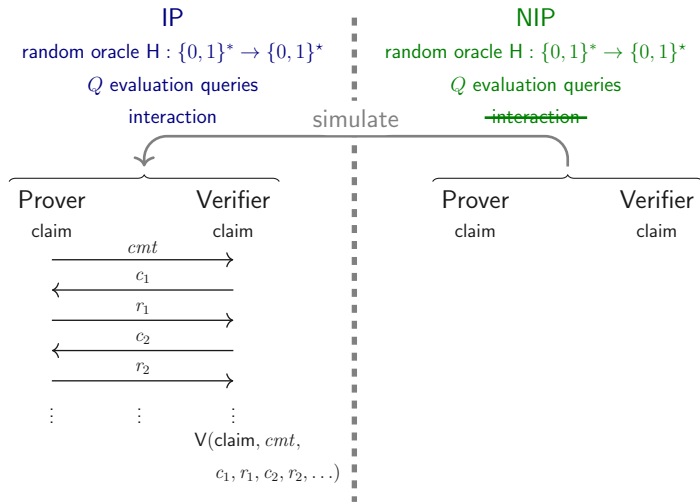
Can remove?

→ Yes! But ...

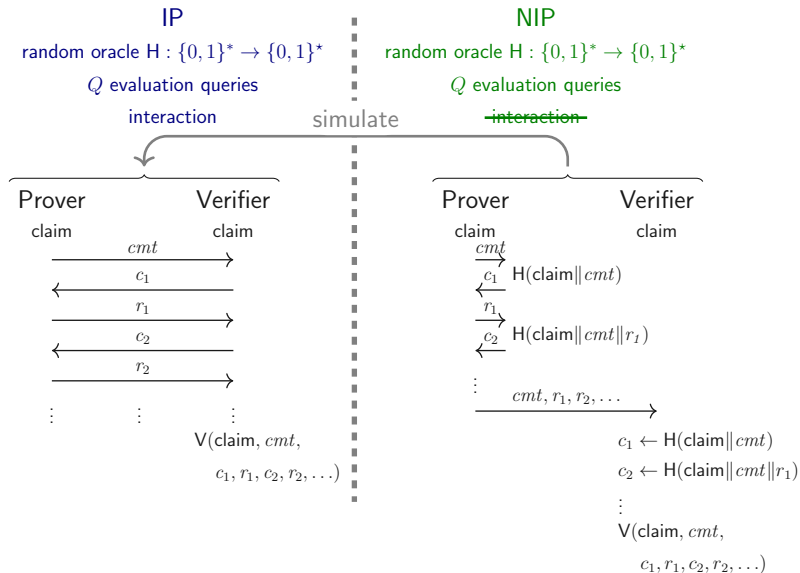
IP to NIP



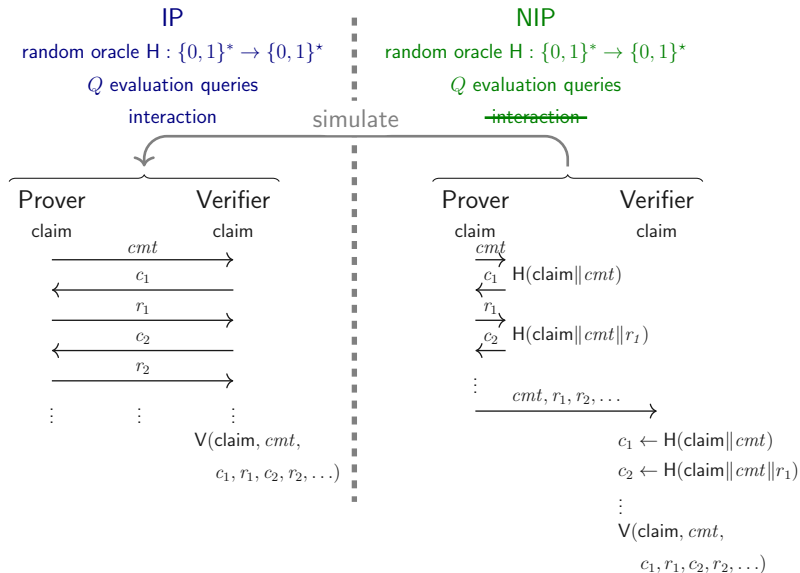
IP to NIP



IP to NIP



IP to NIP



Soundness of Fiat-Shamir: One Round

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

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

Soundness of Fiat-Shamir: One Round

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

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

Soundness of Fiat-Shamir: One Round

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

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 $(x, H(x)) \in \mathcal{R}$

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?

→ depends on # queries

Soundness of Fiat-Shamir: One Round

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

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?

→ 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 $(x, H(x)) \in \mathcal{R}$
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?

→ 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

$H(\text{claim} \parallel cmt) \in \mathcal{S}(cmt)$.

Soundness of Fiat-Shamir: One Round

Task: find x such that $(x, H(x)) \in \mathcal{R}$
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: $\forall x, y : |\{z \mid (x, z) \in \mathcal{R}\}| \approx |\{z \mid (y, z) \in \mathcal{R}\}|$

Apply to (Prover, Verifier):

Find cmt such that

$H(\text{claim} \parallel cmt) \in \mathcal{S}(cmt)$.

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

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

Question: what is the success probability?

→ 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 $(x, H(x)) \in \mathcal{R}$
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?

→ 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

$H(\text{claim} \parallel cmt) \in \mathcal{S}(cmt)$.

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

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

→ soundness error?

→ depends on # queries

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

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

Soundness of Fiat-Shamir: Multiple Rounds

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

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

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

Soundness of Fiat-Shamir: Multiple Rounds

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

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

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

$\mathcal{R}$ is well defined ✓

sparse ✓

uniformity ✗

intuition: “good start” / “bad start”

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

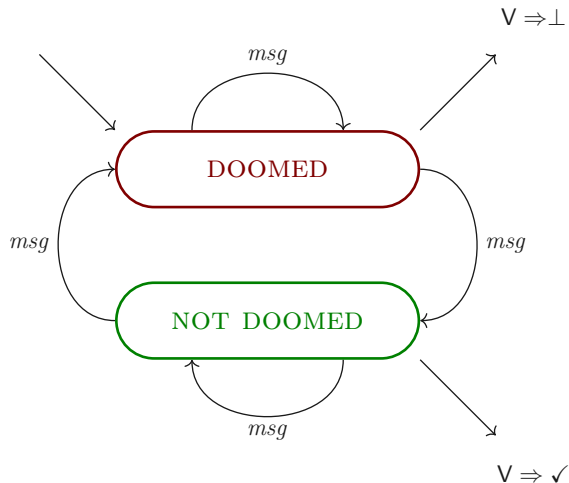
TRANSCRIPT PREFIXES = DOOMED $\sqcup$ NOT DOOMED

DOOMED

NOT DOOMED

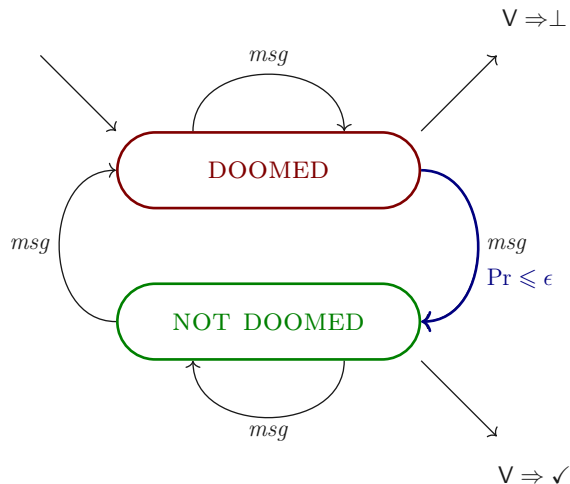
Soundness of Fiat-Shamir: Round-by-Round Soundness

TRANSCRIPT PREFIXES = **DOOMED** $\sqcup$ **NOT DOOMED**



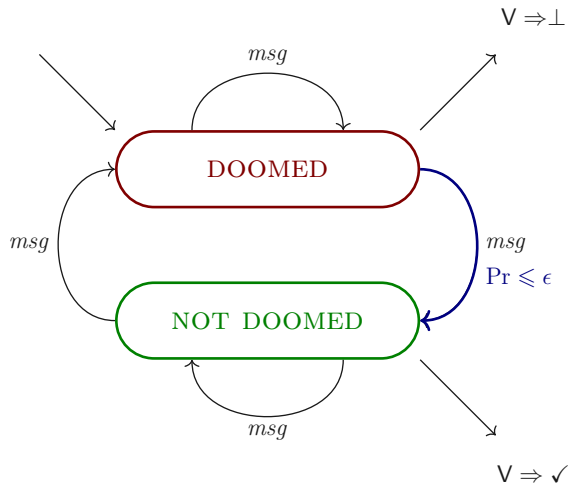
Soundness of Fiat-Shamir: Round-by-Round Soundness

TRANSCRIPT PREFIXES = **DOOMED** $\sqcup$ **NOT DOOMED**



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

TRANSCRIPT PREFIXES = **DOOMED** $\sqcup$ **NOT DOOMED**



- RBR soundness $\Rightarrow$ FS soundness
- RBR soundness of FRI established in [BGKTTZ23]
 - $\rightarrow$ requires *weighted* correlated agreement [BCIKS20]
- RBR soundness of DEEP-ALI established ???
 - $\rightarrow$ specific example: EthSTARK $\checkmark$

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

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

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Table of Contents

Motivation

STARK

Overview

Arithmetization

DEEP-ALI

DEEP

Low Degree Testing

BCS Transform

Fiat-Shamir Transform

Preview

Introduction to STARKs

Alan Szepieniec

艾伦·余丕涅茨

alan@neptune.cash



neptune

<https://neptune.cash/>



Triton VM

<https://triton-vm.org/>

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