Introduction to STARKs and FRI

11 Feb 2025
PSE ZK Workshop @ NUS

overview

Why STARKs?

What is a STARK?

Pipeline of a STARK construction

Exercise: STARK by Hand

adapted from Eran Tromer's "SNARKs with Preprocessing" (2016)

Why STARKs?

ZKPs: Motivation

- f is a computation that takes two weeks to run on a regular computer, but two hours on a data center. You send the data center the computation (ie. the code to run f), the data center runs it, and gives back the answer y with a proof. You verify the proof in a few milliseconds, and are convinced that y actually is the answer.

- You have a blockchain like Ethereum, and you download the most recent block. You want a proof that this block is valid, and that this block is at the tip of a chain where every block in the chain is valid. You ask an existing full node to provide such a proof. x is the entire blockchain (yes, all ?? gigabytes of it), f is a function that processes it block by block, verifies the validity and outputs the hash of the last block, and y is the hash of the block you just downloaded.

STARKs, Part I: Proof with Polynomials by *Vitalik Buterin*

Why STARKs?

1. The T in STARKs stands for ***T****ransparent* - which means it is not reliant on a “trusted setup”
2. Relies on a much simpler cryptographic primitive - hash functions, and some information theoretic constructions
3. As a result of 2) - STARKs are also plausibly quantum resistant

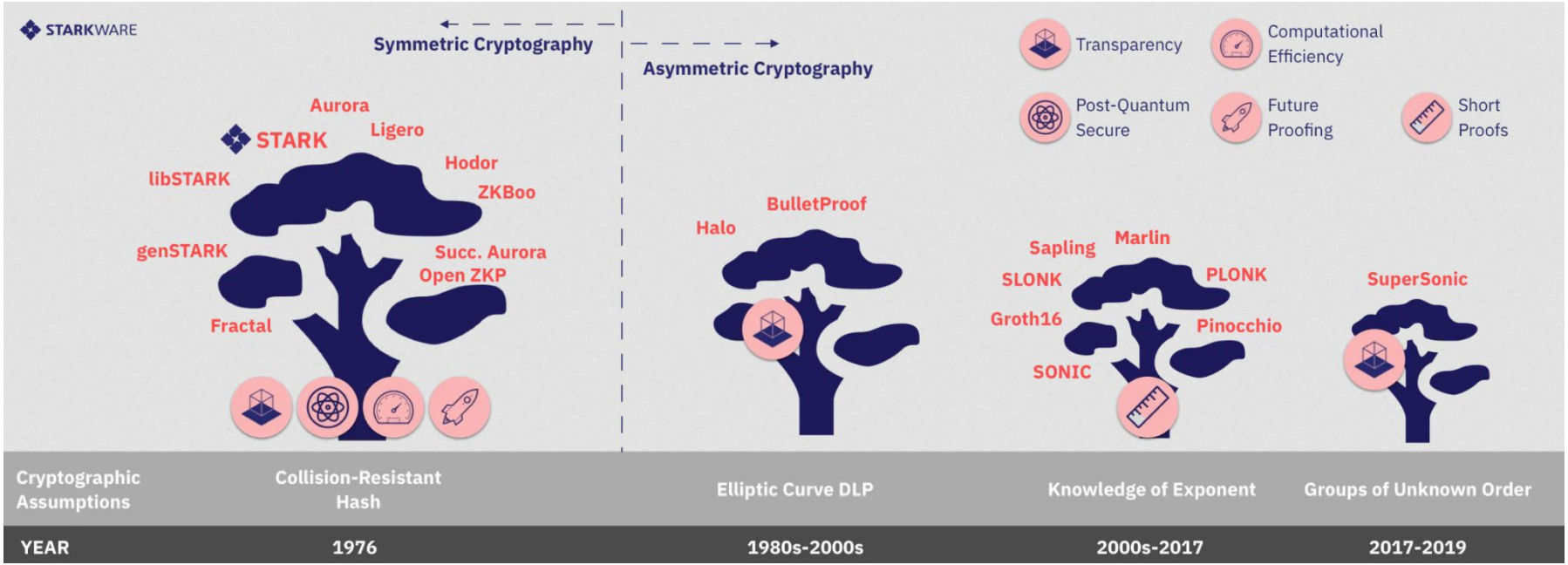
Why STARKs: it is transparent

“Transparent” means it does not rely on a trusted setup (more on this later…)

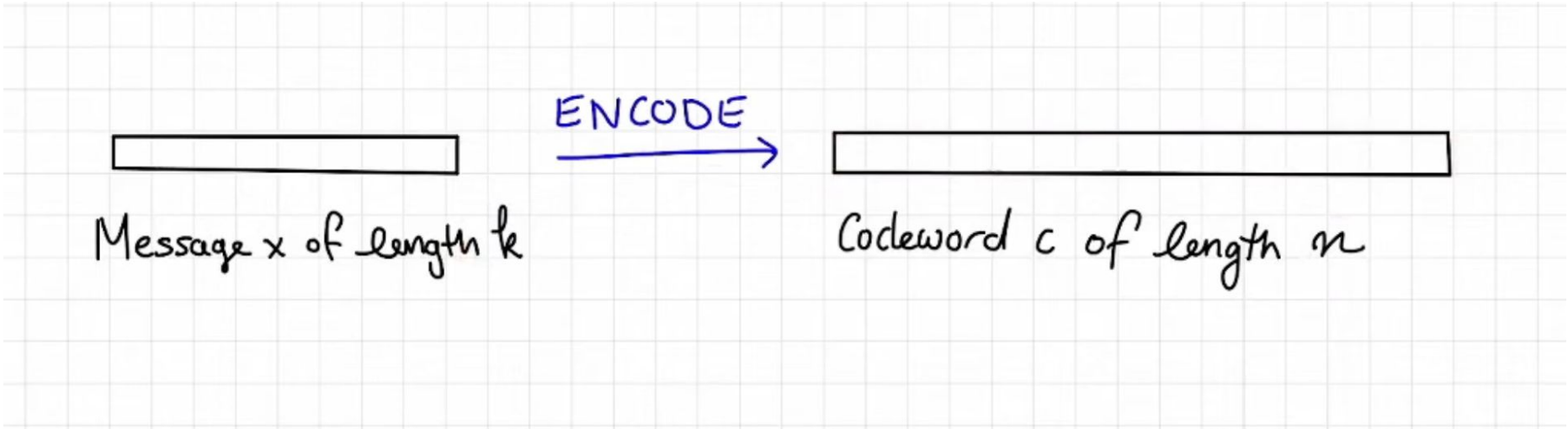
Why STARKs: simpler primitives

- A collision resistant hash function (eg. SHA3, BLAKE3)
- Error correcting codes (in particular, Reed-Solomon codewords)

Why STARKs: simpler primitives



Why STARKs: simpler primitives



adapted from Mary Wootters' 'Introduction to Algebraic Coding Theory' Lecture Series

Why STARKs: (*plausibly)* post quantum

- **Shor's algorithm** breaks elliptic curve cryptography (it efficiently solves the DLP)

- **Grover's algorithm** does not "break" hash functions, but speed up brute-force attacks on them

STARK: Downsides

Proof size in the order of **KBs**, instead of **Bs**

Rough sensing - ***sp1*** (a zkVM) produces proofs of the following sizes:

- Groth16: ~260 bytes
- PLONK: ~868 bytes

source: SP1 Docs

STARKs vs SNARKs?

STARKs use a similar compilation process of arithmetization into a proof system backend

They are also non-interactive and succinct

You can almost think of STARKs as a special case of SNARKs!

What is a STARK?

STARK: History

Scalable, transparent, and post-quantum secure computational integrity

Eli Ben-Sasson* Iddo Bentov† Yinon Horesh* Michael Riabzev*

March 6, 2018

Abstract

Human dignity demands that personal information, like medical and forensic data, be hidden from the public. But veils of secrecy designed to preserve privacy may also be abused to cover up lies and deceit by institutions entrusted with Data, unjustly harming citizens and eroding trust in central institutions.

Scalable, transparent, and post-quantum secure computational integrity (2018)

STARK: History

3.3 Scalable Transparent ARgument of Knowledge (STARK) as a realization of STIK

Definition 3.3 refers to the IOP model, in which results can be proved with no cryptographic assumptions. Indeed, most of our contributions, described in following sections (like the FRI protocol), are stated and studied in this "clean" IOP model; and a majority of our engineering work was dedicated to implementing IOP-based algorithms of a STIK system. However, we are not aware of any unconditionally secure IOP realization that is scalable, and theoretical works show that such constructions are unlikely to emerge [54]. A number of fundamental transformations have been suggested in the past to realize PCP systems using various cryptographic assumptions, and these transformations were adapted to the IOP model [22]. In all such realizations the prover must be computationally bounded, and such systems are commonly called *argument systems*, and, consequently, the realization of a STIK results in a *Scalable Transparent ARgument of Knowledge* (STARK).

Scalable, transparent, and post-quantum secure computational integrity (2018)

STARK: History

3.3 Scalable Transparent ARgument of Knowledge (STARK) as a realization of STIK

the realization of a STIK results in a *Scalable Transparent ARgument of Knowledge* (STARK).

Scalable, transparent, and post-quantum secure computational integrity (2018)

STARK: Definition

the realization of a STIK results in a *Scalable Transparent ARgument of Knowledge* (STARK).

Scalable

- prover runtime: at most quasilinear time in size of computation eg. n log^2 n
- verifier runtime: at most polylog time in size of computation eg. (log n)^2

Transparent

- all verifier messages are public random coins

ARgument of Knowledge

- a computationally sound proof that you know the outcome of some computation

Scalable, transparent, and post-quantum secure computational integrity (2018)

STARK: Definition

the realization of a STIK results in a *Scalable Transparent ARgument of Knowledge* (STARK).

Scalable

- prover runtime: at most quasilinear time in size of computation eg. n log^2 n
- verifier runtime: at most polylog time in size of computation eg. (log n)^2

Transparent

- all verifier messages are public random coins

ARgument of Knowledge

- a computationally sound proof that you know the outcome of some computation

Scalable, transparent, and post-quantum secure computational integrity (2018)

STARKs are *scalable*

Name	Complexity class	Time complexity ($O(n)$)	Examples of running times	Example algorithms
constant time		$O(1)$	10	Finding the median value in a sorted array of numbers. Calculating $(-1)^n$.
logarithmic time	DLOGTIME	$O(\log n)$	$\log n$, $\log(n^2)$	Binary search
polylogarithmic time		$\operatorname{poly}(\log n)$	$(\log n)^2$	
quasilinear time		$n\operatorname{poly}(\log n)$	$n \log^2 n$	Multipoint polynomial evaluation
quadratic time		$O(n^2)$	n^2	Bubble sort. Insertion sort. Direct convolution

https://en.wikipedia.org/wiki/Time_complexity#Table_of_common_time_complexities

STARK: Definition

the realization of a STIK results in a *Scalable Transparent ARgument of Knowledge* (STARK).

Scalable

- prover runtime: at most quasilinear time in size of computation eg. n log^2 n
- verifier runtime: at most polylog time in size of computation eg. (log n)^2

Transparent

- all verifier messages are public random coins

ARgument of Knowledge

- a computationally sound proof that you know the outcome of some computation

Scalable, transparent, and post-quantum secure computational integrity (2018)

STARKs are *transparent*

all verifier messages are public random coins, specifically, random values are derived publicly

Definition 4.2.4. $\mathsf{IARG} = (\mathcal{P}, \mathcal{V})$ *is* **public-coin** *if each message* ρ_i *by the argument verifier* $\mathcal{V}$ *is a random binary string of some prescribed size* r_i*. In this case, the decision bit of the argument verifier* $\mathcal{V}$ *is a function only of the random oracle* f*, the instance* $\mathbb{x}$*, and the transcript of interaction* $(\alpha_1, \rho_1, \ldots, \alpha_{\mathsf{k}}, \rho_{\mathsf{k}})$*; we denote this bit by* $\mathcal{V}^f\left(\mathbb{x}, (\alpha_i)_{i\in[\mathsf{k}]}, (\rho_i)_{i\in[\mathsf{k}]}\right)$.

Definition 4.2.4 from [Building Cryptographic Proofs from Hash Functions (SNARGs book)]

STARK: Definition

the realization of a STIK results in a *Scalable Transparent ARgument of Knowledge* (STARK).

Scalable

- prover runtime: at most quasilinear time in size of computation eg. n log^2 n
- verifier runtime: at most polylog time in size of computation eg. (log n)^2

Transparent

- all verifier messages are public random coins

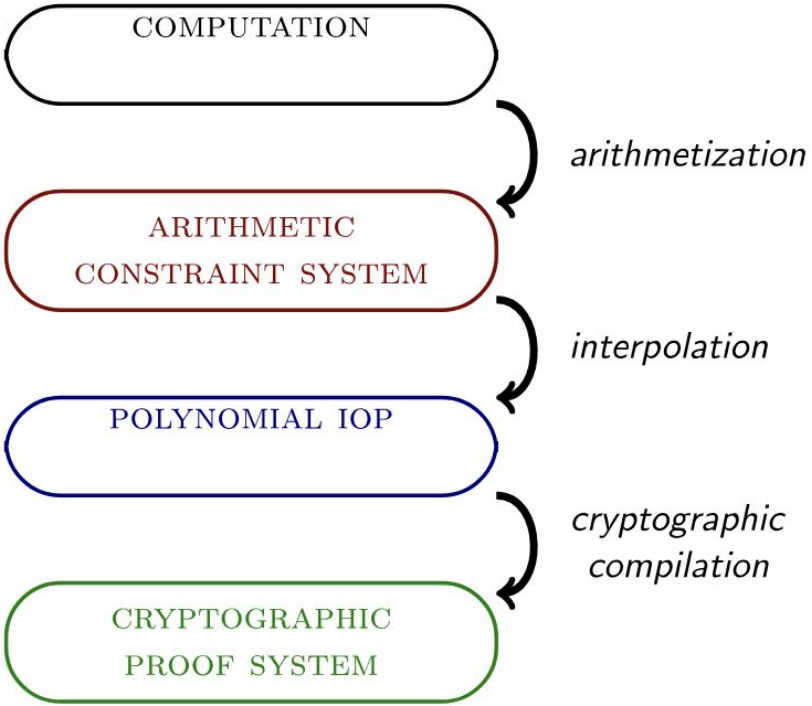
ARgument of Knowledge

- a computationally sound proof that you know the outcome of some computation

Scalable, transparent, and post-quantum secure computational integrity (2018)

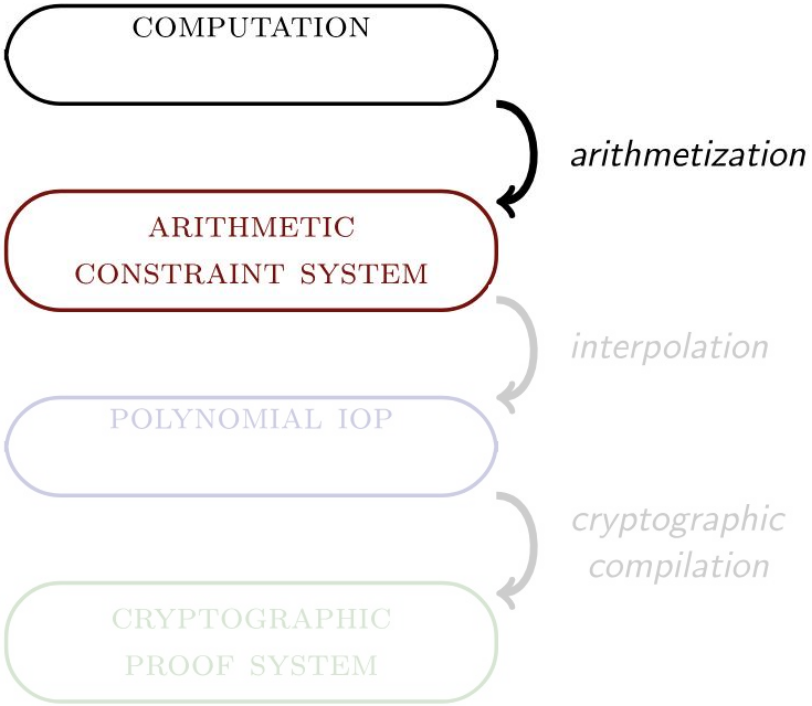
Pipeline of a STARK construction

Pipeline of a STARK construction



Anatomy of a STARK, Part 1: STARK Overview by Alan Szepieniec

Pipeline of a STARK construction



Anatomy of a STARK, Part 1: STARK Overview by Alan Szepieniec

Arithmetization: Motivation

Proof systems do not understand computational integrity statements

eg. “I want to create a proof of a valid transaction on some blockchain”

We need some way to compile such statements into statements that proof systems can understand and reason about

Arithmetization: Goal

Transform a computational integrity statements to statements about systems of low-degree polynomials over finite fields

Concretely, in STARKs, we describe a computation in terms of an ***execution trace*** that satisfies ***constraints*** induced by the correct change in state

This is what is known as an Arithmetic Intermediate Representation (AIR)

AIR: (Informal) Definition

Informally, the AIR is a set

$$\mathcal{P} = \left\{ P_1(\vec{X}, \vec{Y}), \ldots, P_{\mathsf{s}}(\vec{X}, \vec{Y}) \right\}$$

of low degree polynomials with coefficients in $\mathbb{F}$ over a pair of variable sets $\vec{X} = (X_1, \ldots, X_{\mathsf{w}})$ and $\vec{Y} = (Y_1, \ldots, Y_{\mathsf{w}})$ that represent respectively the current and next state of the computation

Scalable, transparent, and post-quantum secure computational integrity (2018)

AIR: (Informal) Definition

Combining

1. State of computation in a tuple of **w** registers, $(t_0, t_1, \ldots, t_w)$, and
2. State transition function: $\mathbb{F}^w \rightarrow \mathbb{F}^w$

forms an Algebraic Execution Trace (henceforth referred to as *execution trace*)

Anatomy of a STARK, Part 1: STARK Overview by Alan Szepieniec

Computation Problem: Fibonacci Sequence

Statement:

“I know 4 steps of the Fibonacci sequence (modulo 97), using 2 user specified inputs.”

AIR: Example

“I know 4 steps of the Fibonacci sequence (modulo 97), using 2 user specified inputs.”

Input 1	Input 2	Output
24	**30**	54
30	54	84
54	84	41
84	41	**28**

AIR: Constraints

To reason about an AIR, we need to define *constraints* on it

We require ***boundary*** and ***transition*** constraints

Boundary constraints:

at the start or end of the computation, an algebraic register has a given value

Transition constraints:

any 2 consecutive state tuples changed in accordance with the STF

AIR: Example

Input 1	Input 2	Output
24	30	54
30	54	84
54	84	41
84	41	28

Boundary constraints:

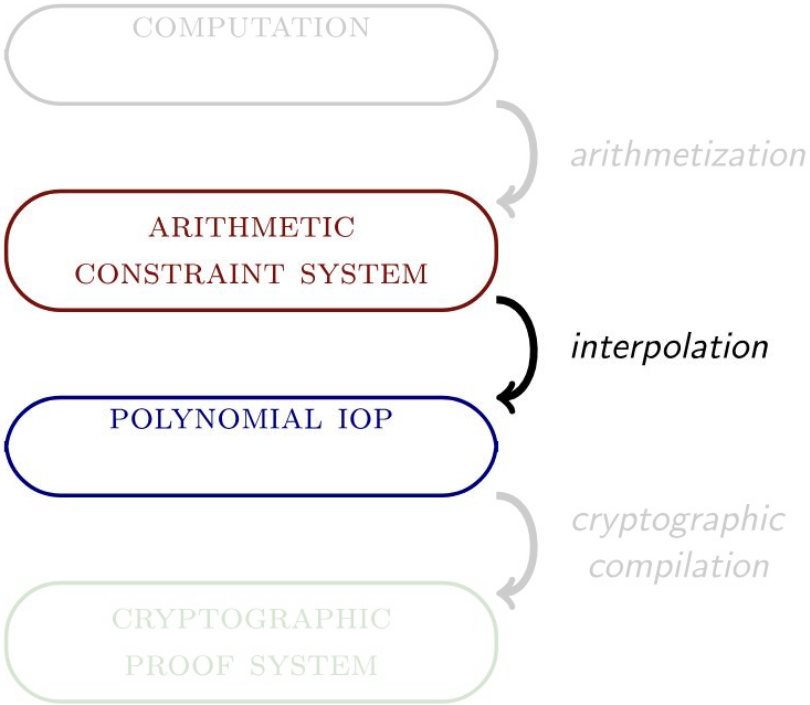
Does user input start with 24 and 30?

Transition constraints:

Do the rows change in accordance with the Fibonacci sequence below?

$$F_n = F_{n-1} + F_{n-2}$$

Pipeline of a STARK construction



Anatomy of a STARK, Part 1: STARK Overview by Alan Szepieniec

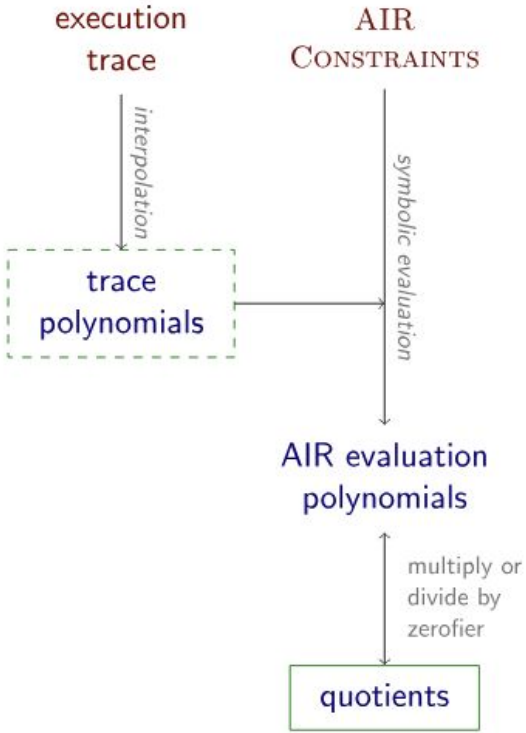
Interpolation

We have our constraint system defined, but we still need further work for our STARK math to be of any use

Represent our *constraint system* in terms of a ***system of polynomials***

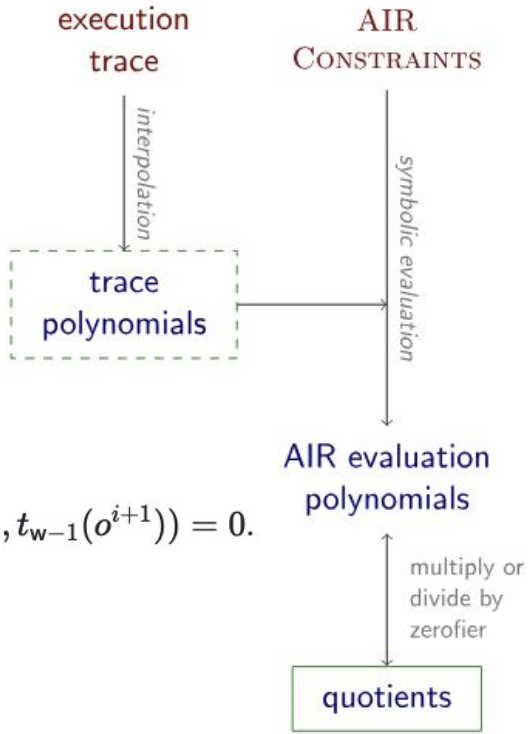
Anatomy of a STARK, Part 4: The STARK IOP *by Alan Szepieniec*

Interpolation



Anatomy of a STARK, Part 4: The STARK IOP by Alan Szepieniec

Interpolation



- All boundary constraints are satisfied: $\forall (i, w, e) \in \mathcal{B} \,.\, t_w(o^i) = e.$
- For all cycles, all transition constraints are satisfied:
  $\forall i \in \{0, \ldots, T-2\} \,.\, \forall j \in \{0, \ldots, r-1\} \,.\, p_j(t_0(o^i), \ldots, t_{\mathsf{w}-1}(o^i), t_0(o^{i+1}), \ldots, t_{\mathsf{w}-1}(o^{i+1})) = 0.$

Anatomy of a STARK, Part 4: The STARK IOP by Alan Szepieniec

Interpolation

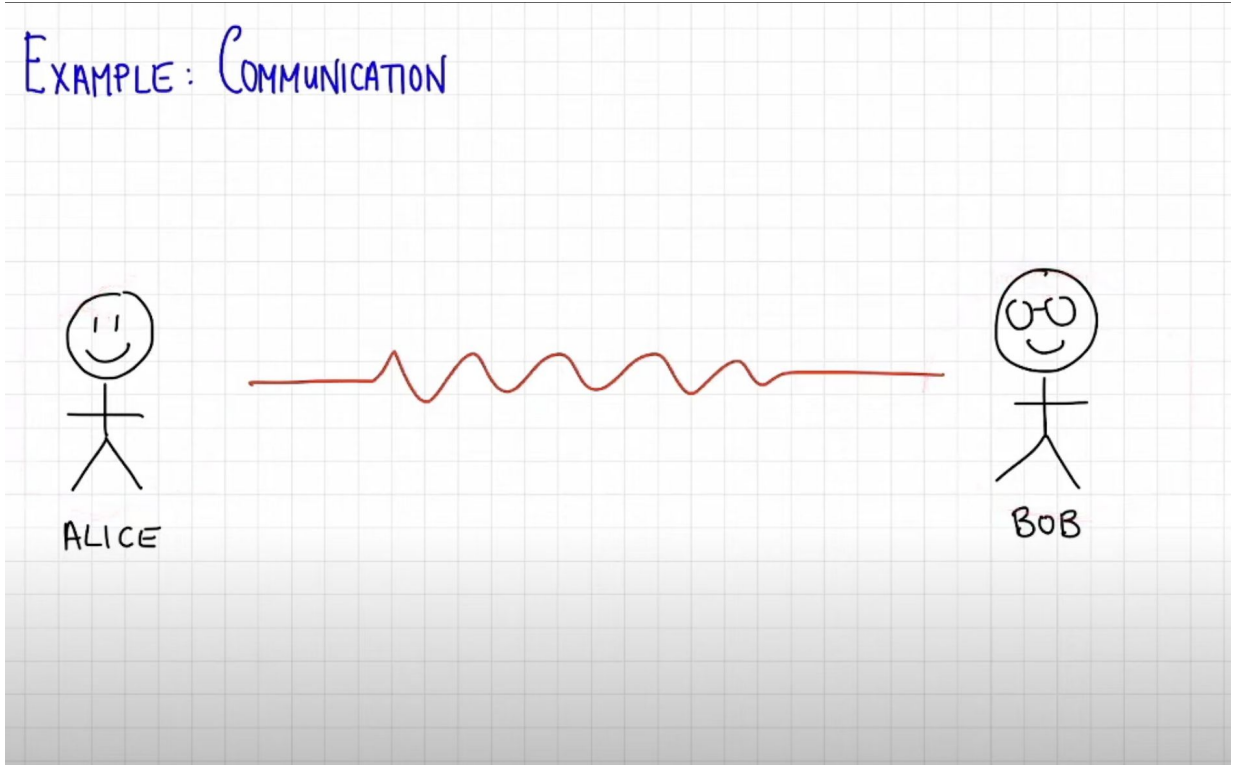
We have our polynomials, but we need to further transform them to make them amenable to the FRI protocol

Compilation via FRI

FRI (***F**ast **R**eed-Solomon **I**nteractive Oracle Proof of Proximity*) is a protocol to establish that a committed polynomial has a bounded degree, using ***codewords***

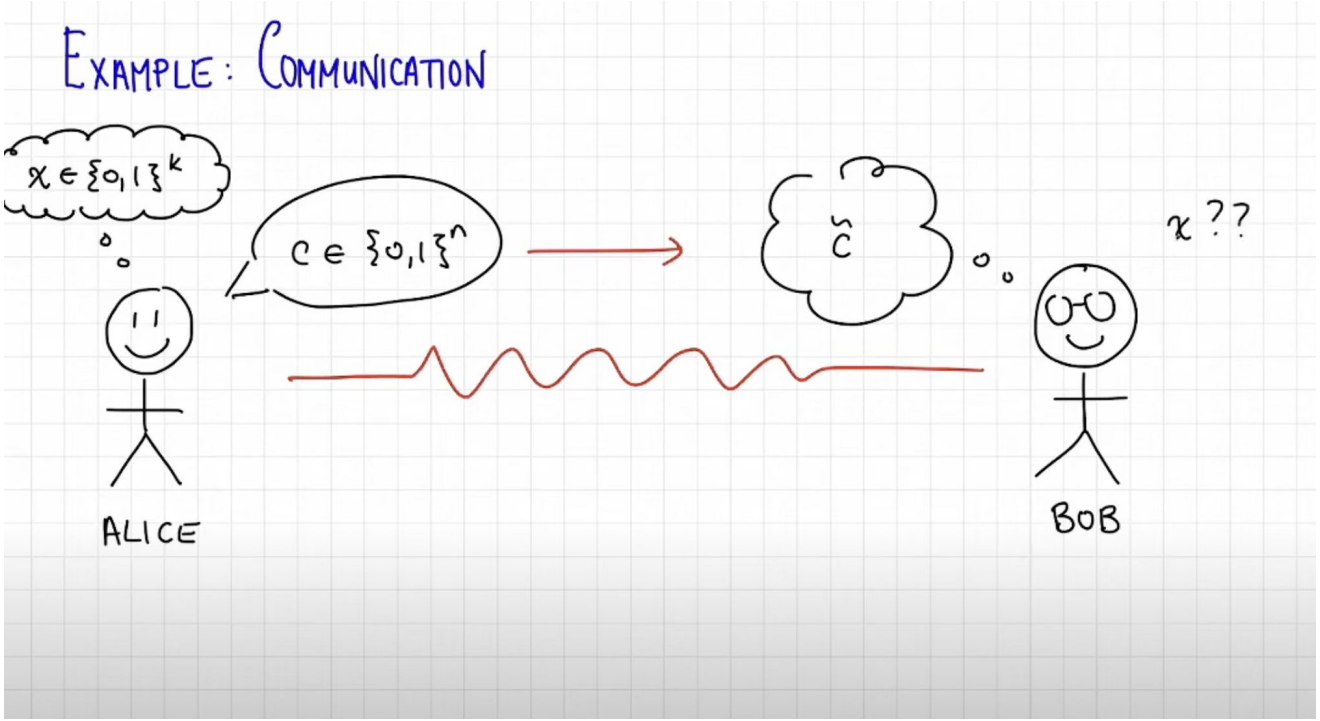
Codewords, in our context, refer to **Reed-Solomon codewords**, a family of *error-correcting codes*

Error-correcting codes: A Quick Primer



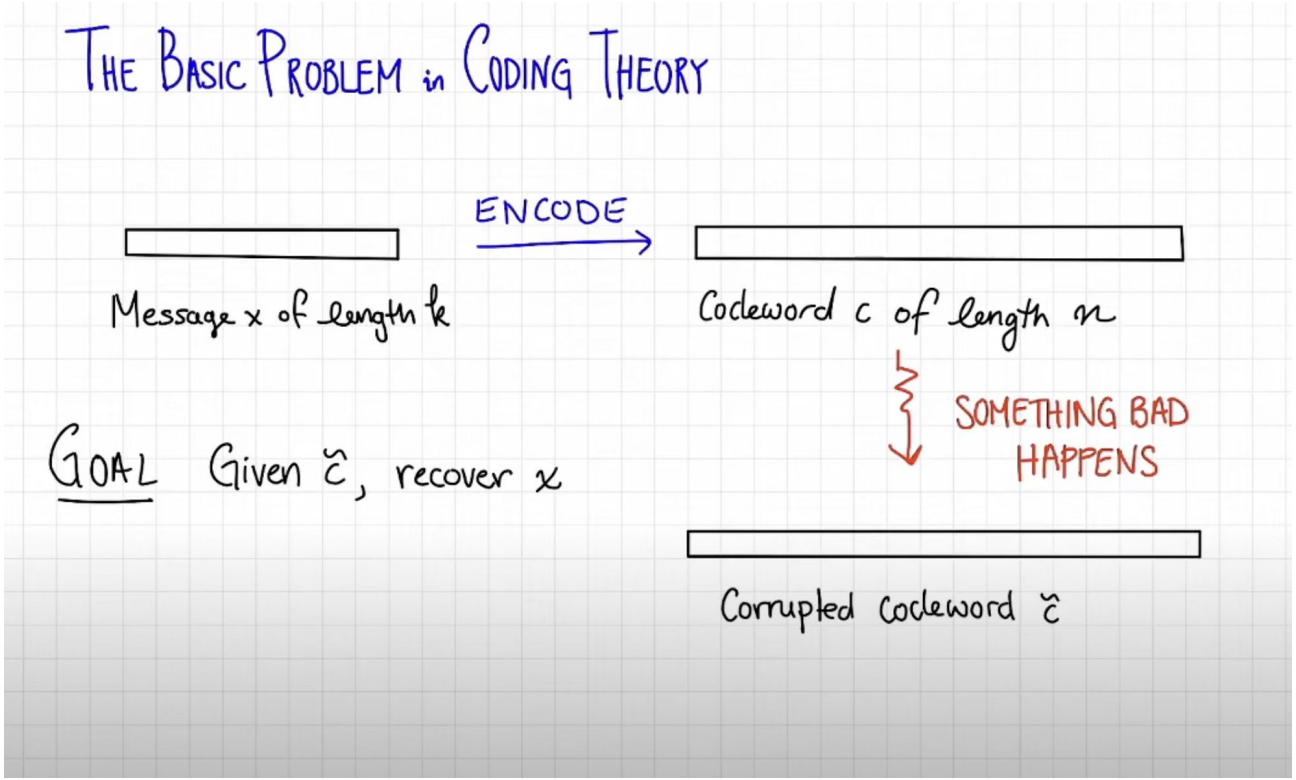
adapted from Mary Wootters' 'Introduction to Algebraic Coding Theory' Lecture Series

Error-correcting codes: A Quick Primer



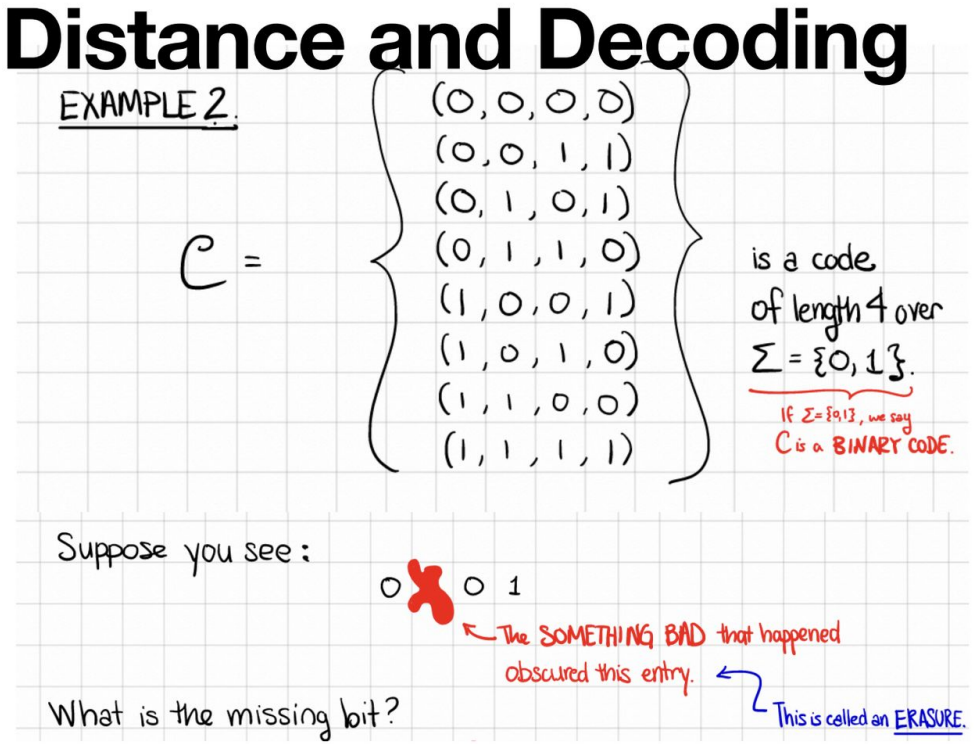
adapted from Mary Wootters' 'Introduction to Algebraic Coding Theory' Lecture Series

Error-correcting codes: A Quick Primer



adapted from Mary Wootters' 'Introduction to Algebraic Coding Theory' Lecture Series

Error-correcting codes: A Quick Primer



Source: Lecture Notes from Lecture 1 of Mary Wootters' course on Algebraic Coding Theory, Winter 2018

adapted from Risc Zero's Intro To FRI

Error-correcting codes: A Quick Primer

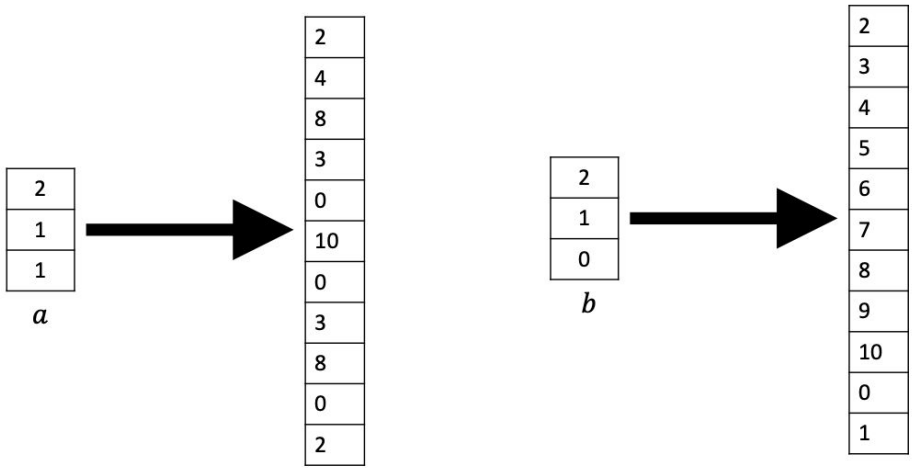
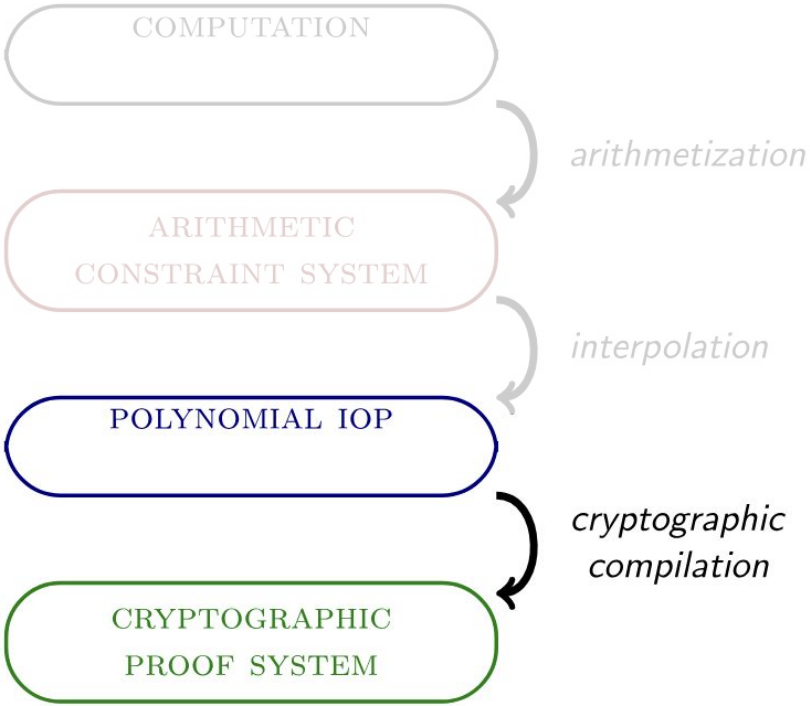


Figure 2.1: On the left is the vector $a = (2,1,1)$ of length 3 with entries interpreted as elements of the field $\mathbb{F}_{11}$, as well as its Reed-Solomon encoding. The Reed-Solomon encoding interprets a as the polynomial $p_a(x) = 2 + x + x^2$ and lists all evaluations of p_a over the field $\mathbb{F}_{11}$. On the right is the vector $b = (2,1,0)$ and its Reed-Solomon encoding.

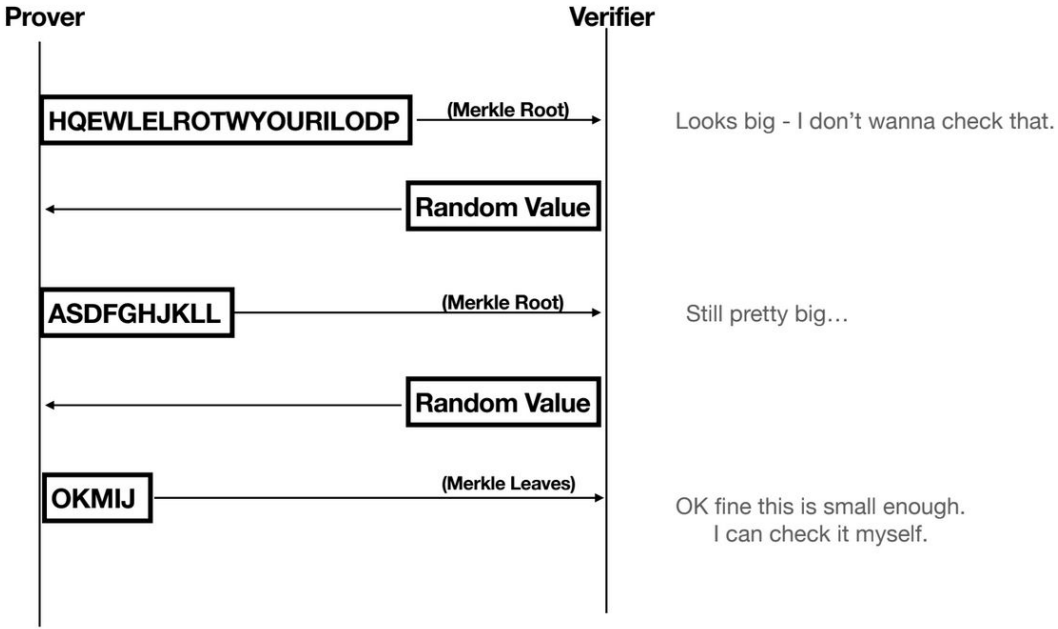
Figure 2.1, Chapter 2 from Proofs, Arguments and Zero-Knowledge by Justin Thaler

Pipeline of a STARK construction



Anatomy of a STARK, Part 1: STARK Overview by Alan Szepieniec

FRI: Overview



FRI in 2 minutes by Paul Gafni (Risc Zero)

FRI Mechanics

See STARK by Hand for a more complete description:
https://www.risczero.com/docs/explainers/proof-system/stark-by-hand#lesson-11-fri-protocol-commit-phase

Begin

Prover sends Merkle root for f_0

Commit Round 1

Verifier sends randomness r_1

Prover **splits** f_0, then **mixes** using r_1

Prover sends Merkle root for f_1

Commit Round 2

Verifier sends randomness r_2

Prover **splits** f_1, then **mixes** using r_2

Prover sends Merkle root for f_2

Each polynomial has half as many coefficients as the previous round.

Each Merkle tree has half as many leaves as the previous round.

Fri Mechanics: Folding, Committing, and Batching by Paul Gafni (Risc Zero)

FRI Mechanics: Splitting

See STARK by Hand for a more complete description:
https://www.risczero.com/docs/explainers/proof-system/stark-by-hand#lesson-11-fri-protocol-commit-phase

RISC ZERO

$$f_0(x) = 19 + 56x + 34x^2 + 48x^3 + 43x^4 + 37x^5 + 10x^6 + 0x^7$$

We've turned one polynomial with 8 coefficients…

…into 2 polynomials with 4 coefficients.

$$= 19 + 34x^2 + 43x^4 + 10x^6 + \longrightarrow f_{0,even}(x) = 19 + 34x + 43x^2 + 10x^3$$

$$56x + 48x^3 + 37x^5 + 0x^7 \longrightarrow f_{0,odd}(x) = 56 + 48x + 37x^2 + 0x^3$$

Fri Mechanics: Folding, Committing, and Batching by Paul Gafni (Risc Zero)

FRI Mechanics: Mixing

See STARK by Hand for a more complete description:
https://www.risczero.com/docs/explainers/proof-system/stark-by-hand#lesson-11-fri-protocol-commit-phase

RISC ZERO

$$f_{0,even}(x) = 19 + 34x + 43x^2 + 10x^3$$
$$f_{0,odd}(x) = 56 + 48x + 37x^2 + 0x^3$$

$19 + 12 \cdot 56$

$34x + 12 \cdot 48x$

Add term-by-term, using randomness

Say $r_1 = 12$

$$f_1 = f_{0,even} + r_1 \cdot f_{0,odd}$$

$$f_1(x) = (19 + 34x + 43x^2 + 10x^3) + 12 \cdot (56 + 48x + 37x^2 + 0x^3)$$

Grouping terms and reducing modulo 97…

$$f_1(x) = 12 + 28x + 2x^2 + 10x^3$$

By **splitting** and then **mixing**,

we have **folded** an 8 coefficient polynomial into a 4 coefficient polynomial.

After 3 rounds, we'd have a constant polynomial!

Fri Mechanics: Folding, Committing, and Batching by Paul Gafni (Risc Zero)

FRI Mechanics: Queries

See STARK by Hand for a more complete description:
https://www.risczero.com/docs/explainers/proof-system/stark-by-hand#lesson-11-fri-protocol-commit-phase

Queries serve as a random challenge.

Blow-up factor of 4 means query has ~3/4 chance of catching cheating prover.

1 query —> 2 bits of security*

* We're hand-waving here; the security story is substantially more complex than this.

Example

Verifier queries at g

Prover sends $f_0(g)$ and $f_0(-g)$

$f_1(g^2)$ and $f_1(-g^2)$

$f_2(g^4)$ and $f_2(-g^4)$

Verifier checks round-by-round consistency:

$f_1(g^2)$ is determined by $f_0(g)$ and $f_0(-g)$

$f_2(g^4)$ is determined by $f_1(g^2)$ and $f_1(-g^2)$

$f_3(g^8)$ is determined by $f_2(g^4)$ and $f_2(-g^4)$

Resources

Papers

Scalable, transparent, and post-quantum secure computational integrity (2018)
A summary on the FRI low degree test

Blog posts, videos

Anatomy of a STARK by Alan Szepieniec

About FRI Protocol by Risc Zero

Intro to FRI: Risc Zero Study Club by Risc Zero

Questions?

Exercise time!