Some important tools for verifiable computation: the Sumcheck protocols

Clémence Chevignard

Overview

1. What is a verifiable computation?
2. Arithmetic circuits and R1CS
3. Reed-Solomon codes
4. The univariate Sumcheck
5. The multivariate Sumchecks
6. Conclusion

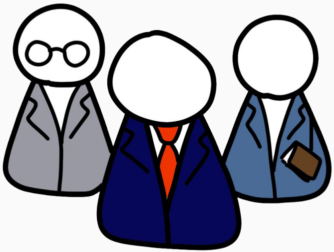
What is a verifiable computation?

What is a verifiable computation?

You

programm + input →

A company X



Small computing capability.

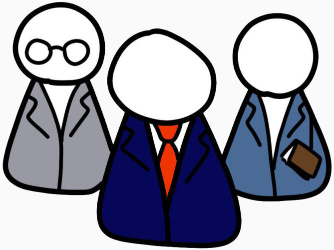
Big computing capability.

What is a verifiable computation?

You

programm + input →

A company X



Small computing capability.

Big computing capability.

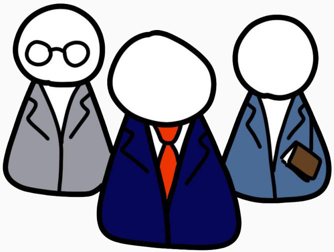
Within 30s you get a result.

What is a verifiable computation?

You

A company X

programm + input →



Small computing capability.

Big computing capability.

Within 30s you get a result.

How to be sure that X's answer is correct?

What is a verifiable computation?

- Solution 1: You do the computation by yourself to check $\to$ not efficient.
- Solution 2: You ask the company X for a proof $\to$ ok, but how?

What is a verifiable computation?

- Solution 1: You do the computation by yourself to check $\rightarrow$ not efficient.
- Solution 2: You ask the company X for a proof $\rightarrow$ ok, but how? $\rightarrow$ a protocol!

What is a verifiable computation?

- Solution 1: You do the computation by yourself to check $\rightarrow$ not efficient.
- Solution 2: You ask the company X for a proof $\rightarrow$ ok, but how? $\rightarrow$ a protocol!

Goals of a verifiable computation protocol:

What is a verifiable computation?

- Solution 1: You do the computation by yourself to check $\rightarrow$ not efficient.
- Solution 2: You ask the company X for a proof $\rightarrow$ ok, but how? $\rightarrow$ a protocol!

Goals of a verifiable computation protocol:

- allowing the company X to give a **proof** that the result is correct.

What is a verifiable computation?

- Solution 1: You do the computation by yourself to check $\rightarrow$ not efficient.
- Solution 2: You ask the company X for a proof $\rightarrow$ ok, but how? $\rightarrow$ a protocol!

Goals of a verifiable computation protocol:

- allowing the company X to give a **proof** that the result is correct.
- . . . without spending **more time to craft the proof than to do the computation**.

What is a verifiable computation?

- Solution 1: You do the computation by yourself to check $\rightarrow$ not efficient.
- Solution 2: You ask the company X for a proof $\rightarrow$ ok, but how? $\rightarrow$ a protocol!

Goals of a verifiable computation protocol:

- allowing the company X to give a **proof** that the result is correct.
- ... without spending **more time to craft the proof than to do the computation**.
- You must be able to check the proof **faster than doing the computation**.

Terminology

You $\rightarrow$ the "Verifier"

The company X $\rightarrow$ the "Prover"

Terminology

You $\to$ the "Verifier"

The company X $\to$ the "Prover"

One type of protocols model $\to$ **IOP model:** *Interactive Oracle Proof* [BCS16]:

- allows V and P interactions: they can send each other messages during several rounds.
- allows V to have oracle access to P's messages.
- V can use randomness to make queries to P's oracles.

Terminology

You $\rightarrow$ the "Verifier"

The company X $\rightarrow$ the "Prover"

One type of protocols model $\rightarrow$ **IOP model:** *Interactive Oracle Proof* [BCS16]:

- allows V and P interactions: they can send each other messages during several rounds.
- allows V to have oracle access to P's messages.
- V can use randomness to make queries to P's oracles.

The oracle notion is theoretical, but can be implemented with Merkle trees.

Some precisions

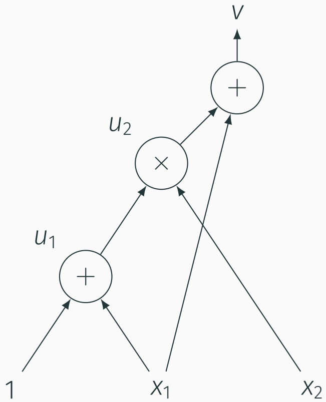
What do we need to be careful about:

- Completeness.
- Linear Prover time.
- Sublinear Verifier time.
- Soundness.

- Linear Proof length, or less.
  proof length = total length of prover's oracles.
- Sublinear Query complexity.
  query complexity = elements read by the Verifier.

Arithmetic circuits and R1CS

What's an **arithmetic circuit** C?



Represents the computation

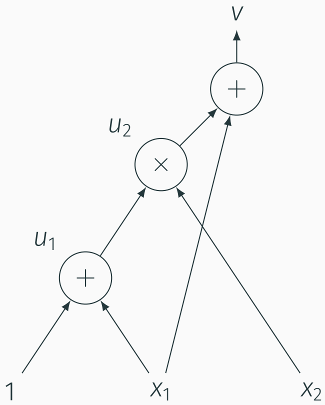
$$v = (1 + x_1) \times x_2 + x_1.$$

Claim: $C(1, x_1, x_2) = v$.

x_1 and x_2 are the inputs of the circuit, v is the output. Every variable belongs to $\mathbb{F}$.

"Length of the computation" $= |(1, x_1, x_2, u_1, u_2, v)|$.

We can build a **Rank 1 Constraint Satisfiability** (A, B, C, x, v) from it.



$\Rightarrow$

$$Az \odot Bz = Cz \text{ with}$$

$$z^T = (1, x_1, x_2, u_1, u_2, v)$$

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

where “ $\odot$ ” is a coefficient-wise product.

Claim that $C(1, x_1, x_2) = v \Leftrightarrow \exists (u_1, u_2, v) / Az \odot Bz = Cz$.

We can build a **Rank 1 Constraint Satisfiability** (A, B, C, x, v) from it.

Why does it work?

Line 1 of A, B, C:

- $(Az)_1 = 1 + x_1$
- $(Bz)_1 = 1$
- $(Cz)_1 = u_1 = 1 \times (1 + x_1)$

$$Az \odot Bz = Cz \text{ with}$$

$$z^T = (1, x_1, x_2, u_1, u_2, v)$$

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

where “ $\odot$ ” is a coefficient-wise product.

From now on the goal of P is to prove to V that it exists u_1, u_2, v such that $Az \odot Bz = Cz$.

We can build a **Rank 1 Constraint Satisfiability** (A, B, C, x, v) from it.

$$Az \odot Bz = Cz \text{ with}$$

$$z^T = (1, x_1, x_2, u_1, u_2, v)$$

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

where “ $\odot$ ” is a coefficient-wise product.

Why does it work?

Line 1 of A, B, C:

- $(Az)_1 = 1 + x_1$
- $(Bz)_1 = 1$
- $(Cz)_1 = u_1 = 1 \times (1 + x_1)$

From now on the goal of P is to prove to V that it exists u_1, u_2, v such that $Az \odot Bz = Cz$.

Still a bit vague, let’s make it more precise.

R1CS

P knows $z = (1, x_1, \ldots, x_n, u_1, \ldots, u_{n'}, v) = (1||x||u||v)$, with $u = (u_1, \ldots, u_{n'})$ supposed to be the outputs of the gates of the circuit.

V knows $(1||x)$ and v.

R1CS [BCRSVW18]

A R1CS instance is specified by $n \times m$ matrices A, B, C over $\mathbb{F}$ and by a vector x and an element v over $\mathbb{F}$.

It is satisfied by a vector u if and only if $Az \odot Bz = Cz$, $z := (1||x||u||v)$.

$\rightarrow$ the whole instance $= (\mathbb{F}, n, m, A, B, C, x, v)$.

P knows $z = (1, x_1, \ldots, x_n, u_1, \ldots, u_{n'}, v) = (1||x||u||v)$, with $u = (u_1, \ldots, u_{n'})$ supposed to be the outputs of the gates of the circuit.

V knows $(1||x)$ and v.

R1CS [BCRSVW18]

A R1CS instance is specified by $n \times m$ matrices A, B, C over $\mathbb{F}$ and by a vector x and an element v over $\mathbb{F}$.

It is satisfied by a vector u if and only if $Az \odot Bz = Cz$, $z := (1||x||u||v)$.

$\rightarrow$ the whole instance $= (\mathbb{F}, n, m, A, B, C, x, v)$.

The relation R_{R1CS} is the set of tuples $((\mathbb{F}, n, m, A, B, C, x, v), u)$ such that u satisfies $(\mathbb{F}, n, m, A, B, C, x, v)$.

The Univariate Sumcheck

The “Aurora” [BCRSVW18] article proposes a protocol for R1CS relations

- Define $z_A = Az, z_B = Bz, z_C = Cz$.
- Separately check that $Az = z_A$, $Bz = z_B$, $Cz = z_C \rightarrow$ lincheck.
- Then check that $z_A \odot z_B = z_C \rightarrow$ rowcheck.

A core non-trivial ingredient is to be able to check the statement

$$\sum_{a \in H} \hat{f}(a) = \mu,$$

given $H \subset \mathbb{F}$ with $|H| =$ number of variables, $\hat{f}(X) \in \mathbb{F}[X]$, $\mu \in \mathbb{F}$.

The univariate sumcheck is a protocol that allows to do so.

The Univariate Sumcheck

We need, on input $H \subset \mathbb{F}$, $\hat{f}(X) \in \mathbb{F}[X]$, $\mu \in \mathbb{F}$, to be able to check that

$$\sum_{a \in H} \hat{f}(a) = \mu$$

The **univariate sumcheck** is a protocol that allows to do so.

Why not simply computing the sum?

- $O(|H|)$ evaluations of $\hat{f}(X)$ for the Verifier.
- An evaluation of $\hat{f}(X)$ costs $O(\deg \hat{f}(X))$ operations.

$\rightarrow$ way too long!

Reed-Solomon codes

Reed-Solomon codes

Given $L \subset \mathbb{F}$, $0 < d \leqslant |L|$, we denote by $RS[L, d]$ the evaluations over L of all polynomials of $\mathbb{F}[X]$ of degree $< d$.

Reed-Solomon codes

Given $L \subset \mathbb{F}$, $0 < d \leqslant |L|$, we denote by $RS[L, d]$ the evaluations over L of all polynomials of $\mathbb{F}[X]$ of degree $< d$.

Encoding of a vector t into a codeword

Define $H = \{h_1, \ldots, h_d\}, L = \{\ell_1, \ldots, \ell_n\} \subset \mathbb{F}$ such that $|H| \leqslant |L|$, and $t \in \mathbb{F}^{|H|}$:

1. The "low degree extension" $\hat{f}_t(X)$ of t is defined as the **only polynomial of degree** $< |H|$ such that
$$\forall i \in \{1, \ldots, d\}, \hat{f}_t(h_i) = t_i.$$
2. $f_t := \hat{f}_t|_L := (f_t(\ell_1), \ldots, f_t(\ell_n))$ **is the codeword that encodes** t.

The FRI

What are we going to do with RS codewords?

1. Compute $\hat{f}_t(X)$ from H and t.
2. "**Check, given a vector f_t, that f_t belongs to** $RS[L, d]$."
   $\rightarrow$ Low degree test FRI [BBHR17]: Fast Reed-Solomon Interactive oracle proof of proximity.

The FRI

What are we going to do with RS codewords?

1. Compute $\hat{f}_t(X)$ from H and t.
2. "**Check, given a vector f_t, that f_t belongs to** $RS[L, d]$."
   $\rightarrow$ **Low degree test FRI [BBHR17]:** Fast Reed-Solomon Interactive oracle proof of proximity.

FRI $=$ **IOPP**, Interactive Oracle Proof of **Proximity**

- Allows interactions, oracle access, randomness ...
- Locality: logarithmic number of query.
- "Proximity" $\rightarrow$ the protocol checks **whether a vector f_t is in** $RS[L, d]$ **(so in** $RS[L, d]$ **with a certain probability) or far from it**.

The FRI, if L is well choosen, has the performance:

- *Prover time* < $6|L|$.
- *Verifier time* $\leqslant 21 \log |L|$.
- *Proof length* $< |L|/3$.
- *Query complexity* $= 2 \log |L|$.

The univariate Sumcheck

The univariate Sumcheck

Sumcheck Relation

The relation R_{SUM} is the set of all pairs $((\mathbb{F}, L, H, d, \mu), f_t)$ where

- $L, H \subset \mathbb{F}$
- $0 < d < |L|$
- $\mu \in \mathbb{F}$
- $f_t \in RS[L, d]$
- $\sum_{a \in H} \hat{f}_t(a) = \mu$.

The univariate Sumcheck

Sumcheck Relation

The relation R_{SUM} is the set of all pairs $((\mathbb{F}, L, H, d, \mu), f_t)$ where

- $L, H \subset \mathbb{F}$
- $0 < d < |L|$
- $\mu \in \mathbb{F}$
- $f_t \in RS[L, d]$
- $\sum_{a \in H} \hat{f}_t(a) = \mu$.

We can make an IOP protocol for the Sumcheck relation.

The univariate Sumcheck

A useful result

If H is an additive subgroup of $\mathbb{F}$, given a polynomial $\hat{g}(X)$ such that $\deg \hat{g}(X) \leqslant |H| - 1$ and the coefficient of $X^{|H|-1}$ in $\hat{g}(X)$ is α, we have

$$\sum_{a \in H} \hat{g}(a) = \alpha \sum_{a \in H} a^{|H|-1}.$$

The univariate Sumcheck

Setup/Inputs of the sumcheck: P knows f, V has oracle access to f.

Claim: $\sum_{a \in H} \hat{f}(a) = \mu$

The univariate Sumcheck

Setup/Inputs of the sumcheck: P knows f, V has oracle access to f.

Claim: $\sum_{a \in H} \hat{f}(a) = \mu$

Protocol

1. P computes $\hat{f}(X)$.

The univariate Sumcheck

Setup/Inputs of the sumcheck: P knows f, V has oracle access to f.

Claim: $\sum_{a \in H} \hat{f}(a) = \mu$

Protocol

1. P computes $\hat{f}(X)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.

The univariate Sumcheck

Setup/Inputs of the sumcheck: P knows f, V has oracle access to f.

Claim: $\sum_{a \in H} \hat{f}(a) = \mu$

Protocol

1. P computes $\hat{f}(X)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.
3. P computes $\widehat{g}(X)$ and $\widehat{f}(X)$ such that

$$\hat{f}(X) = \hat{g}(X) + Z_H(X)\hat{h}(X), \text{ deg } \widehat{g}(X) < |H|.$$

The univariate Sumcheck

Setup/Inputs of the sumcheck: P knows f, V has oracle access to f.

Claim: $\sum_{a \in H} \hat{f}(a) = \mu$

Protocol

1. P computes $\hat{f}(X)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.
3. P computes $\widehat{g}(X)$ and $\widehat{f}(X)$ such that
$$\hat{f}(X) = \hat{g}(X) + Z_H(X)\hat{h}(X), \deg \widehat{g}(X) < |H|.$$
4. P gives oracle access to V to $h = \hat{h}|_L$.

The univariate Sumcheck

Setup/Inputs of the sumcheck: P knows f, V has oracle access to f.

Claim: $\sum_{a \in H} \hat{f}(a) = \mu$

Protocol

1. P computes $\hat{f}(X)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.
3. P computes $\widehat{g}(X)$ and $\widehat{f}(X)$ such that
$$\hat{f}(X) = \hat{g}(X) + Z_H(X)\hat{h}(X), \deg \widehat{g}(X) < |H|.$$
4. P gives oracle access to V to $h = \hat{h}|_L$.
5. V and P computes $\zeta = \sum_{a \in H} a^{|H|-1}$.

The univariate Sumcheck

Setup/Inputs of the sumcheck: P knows f, V has oracle access to f.

Claim: $\sum_{a \in H} \hat{f}(a) = \mu$

Protocol

1. P computes $\hat{f}(X)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.
3. P computes $\widehat{g}(X)$ and $\widehat{f}(X)$ such that
$$\hat{f}(X) = \hat{g}(X) + Z_H(X)\hat{h}(X), \deg \widehat{g}(X) < |H|.$$
4. P gives oracle access to V to $h = \hat{h}|_L$.
5. V and P computes $\zeta = \sum_{a \in H} a^{|H|-1}$.
6. V and P **run a FRI protocol** with P to check that
$$p := (\zeta\hat{f}(X) - \zeta Z_H(X)\hat{h}(X) - \mu X^{|H|-1})|_L \in RS[L, |H| - 1],$$

The univariate Sumcheck

Setup/Inputs of the sumcheck: P knows f, V has oracle access to f.

Claim: $\sum_{a\in H} \hat{f}(a) = \mu$

Protocol

1. P computes $\hat{f}(X)$.
2. P and V compute $Z_H(X) = \prod_{a\in H}(X - a)$.
3. P computes $\widehat{g}(X)$ and $\widehat{f}(X)$ such that
$$\hat{f}(X) = \hat{g}(X) + Z_H(X)\hat{h}(X), \deg \widehat{g}(X) < |H|.$$
4. P gives oracle access to V to $h = \hat{h}|_L$.
5. V and P computes $\zeta = \sum_{a\in H} a^{|H|-1}$.
6. V and P **run a FRI protocol** with P to check that
$$p := (\zeta\hat{f}(X) - \zeta Z_H(X)\hat{h}(X) - \mu X^{|H|-1})|_L \in RS[L, |H| - 1],$$
7. **and another** to check that
$$h \in RS[L, \deg \hat{f}(X) - |H| + 1].$$

The univariate Sumcheck

Setup/Inputs of the sumcheck: *P knows f, V has oracle access to f.*

Performance

- *Prover time:*
  - one *IFFT* to get $\widehat{f}(X)$ from f.
  - one "divide-and-conquer" algorithms to get $Z_H(X)$: $O(\log|H|)$.
  - one polynomial divisions to compute $\hat{g}$, $\hat{h}$: $O(M(d))$ $(d = \deg \hat{f}(X))$.
  - two *FFT* to evaluate $\hat{h}(X), \hat{g}(X)$ over L.
  - two FRI: $< 6|L|$.

  so Prover time in $O(M(d)) + 3FFT(\mathbb{F}, L) + 12|L|$.

- *Verifier time:* $O(\log^2|H|)$ (computing ζ) $+42\log|L|)$ (FRI).
- *Query complexity:* $4\log|L|$ related to the low degree test.
- *Proof length:* $2|L|/3$.

Our goal

Sarah, Jade and Daniel made an efficient multivariate FRI recently, for tensor product of Reed-Solomon codes $RS[L_1, d_1] \otimes \ldots \otimes RS[L_n, d_n]$.

Tensor product of RS codes

Given $L_1, \ldots, L_n \subset \mathbb{F}$, $0 < d_1, \ldots, d_n < |L_1|,_d otsc, |L_n|$, we denote by $RS[L_1, d_1] \otimes \ldots \otimes RS[L_n, d_n]$ the evaluations over $L_1 \times \ldots \times L_n$ of all polynomials $\hat{f}(X_1, \ldots, X_n)$ of $\mathbb{F}[X_1, \ldots, X_n]$ such that $\forall i \in \{1, \ldots, n\}$, $\deg_{X_i} \hat{f}(X_1, \ldots, X_n) < d_i$.

Sarah, Jade and Daniel made an efficient multivariate FRI recently, for tensor product of Reed-Solomon codes $RS[L_1, d_1] \otimes \ldots \otimes RS[L_n, d_n]$.

Tensor product of RS codes

Given $L_1, \ldots, L_n \subset \mathbb{F}$, $0 < d_1, \ldots, d_n < |L_1|,_d otsc, |L_n|$, we denote by $RS[L_1, d_1] \otimes \ldots \otimes RS[L_n, d_n]$ the evaluations over $L_1 \times \ldots \times L_n$ of all polynomials $\hat{f}(X_1, \ldots, X_n)$ of $\mathbb{F}[X_1, \ldots, X_n]$ such that $\forall i \in \{1, \ldots, n\}$, $\deg_{X_i} \hat{f}(X_1, \ldots, X_n) < d_i$.

$\rightarrow$ Let's use it to build a multivariate Sumcheck.

Our goal

$\rightarrow$ Let's use it to build a multivariate Sumcheck.

... **Actually, this already exists.**

Our goal

$\rightarrow$ Let's use it to build a multivariate Sumcheck.

. . . **Actually, this already exists.**

The first multivariate sumcheck is from Carsten Lund et al [LFKN90] and was related to the SAT and UNSAT problems.

In fact, it has many applications.

Multivariate Sumcheck

IP protocol: Interactive protocol, with V reading all the messages it receives.

Inputs: P knows $\hat{p}(X_1, \ldots, X_n)$, V has oracle access to $\hat{p}(X_1, \ldots, X_n)$ and its degree.

Claim: $\sum_{a_1,\ldots,a_n \in H} \hat{p}(a_1, \ldots, a_n) = \alpha$.

IP protocol: Interactive protocol, with V reading all the messages it receives.

Inputs: P knows $\hat{p}(X_1,\ldots,X_n)$, V has oracle access to $\hat{p}(X_1,\ldots,X_n)$ and its degree.

Claim: $\sum_{a_1,\ldots,a_n\in H}\hat{p}(a_1,\ldots,a_n)=\alpha$.

Protocol

Prover		Verifier
$\hat{p}_1(X) := \sum_{a_2,\ldots,a_n\in H}\hat{p}(X,a_2,\ldots,a_n)$	$\xrightarrow{\hat{p}_1(X)}$	$\sum_{a_1\in H}\hat{p}_1(a_1)\stackrel{?}{=}\alpha$
	$\xleftarrow{w_1}$	$w_1\stackrel{\$}{\leftarrow}\mathbb{F}$
$\hat{p}_2(X) := \sum_{a_3,\ldots,a_n\in H}\hat{p}(w_1,X,a_3,\ldots,a_n)$	$\xrightarrow{\hat{p}_2(X)}$	$\sum_{a_2\in H}\hat{p}_2(a_2)\stackrel{?}{=}\hat{p}_1(w_1)$
$\vdots$	$\vdots$	$\vdots$
$\hat{p}_n(X) := \hat{p}(w_1,\ldots,w_{n-1},X)$	$\xrightarrow{\hat{p}_n(X)}$	$\sum_{a_n\in H}\hat{p}_n(a_n)\stackrel{?}{=}\hat{p}_{n-1}(w_{n-1})$
		$w_n\stackrel{\$}{\leftarrow}\mathbb{F}$
		$\hat{p}(w_1,w_2,\ldots,w_n)\stackrel{?}{=}\hat{p}_n(w_n)$

Multivariate Sumcheck

Performance

- *Prover time:* $|H|^n$.
- *Verifier time:* $n|H| \deg_{ind} \hat{p}(X_1, \ldots, X_n)$.
- *Communication cost:* $n \deg_{ind} \hat{p}(X_1, \ldots, X_n)$.

Multivariate Sumcheck - an improvement

Original Multivariate Sumcheck $\leftarrow$ [LFKN90].

Ben-Sasson et al [BCGRS17] proposed an alternative algorithm, **using a univariate sumcheck**, **Reed-Solomon codes**, and a big abstract theorem [MIE09], to have better computing time.

Multivariate Sumcheck - an improvement

Original Multivariate Sumcheck $\leftarrow$ [LFKN90].

Ben-Sasson et al [BCGRS17] proposed an alternative algorithm, **using a univariate sumcheck**, **Reed-Solomon codes**, and a big abstract theorem [MIE09], to have better computing time.

[BCGRS17]

- *Prover times:* $npoly(\log|\mathbb{F}|) + nO(|L|^2 + |H|\log(|L|^2 + |H|)) + n|L|^n$.
- *Verifier times:* $n \times poly(\log|\mathbb{F}| + \log(|L|^2 + |H|)) + O(n)$.
- *Proof length:* $O(n(|L|^2 + |H|)\log(|L|^2 + |H|))$.
- *Query complexity:* $O(n)$.

[LFKN90]]

- *Prover time:* $|H|^n$.
- *Verifier time:* $n|H| \deg_{ind} \hat{p}(X_1, \ldots, X_n)$.
- *Communication cost:* $n \deg_{ind} \hat{p}(X_1, \ldots, X_n)$.

Much better!

Can we do better ?

My multivariate Sumcheck

Let's focus on the bivariate case:

Another useful result

If H is an additive subgroup of $\mathbb{F}$, given a polynomial $\hat{f}(X, Y)$ such that $\deg_{X,Y} \hat{f} \leqslant |H| - 1$ and α is the coefficient of $X^{|H|-1}Y^{|H|-1}$ in $\hat{f}$, we have

$$\sum_{a_1, a_2 \in H} \hat{f}(a_1, a_2) = \alpha \sum_{a_1, a_2 \in H} a_1^{|H|-1} a_2^{|H|-1}.$$

Inputs: P knows $f = \hat{f}|_{L \times L}$, V has oracle access to $f = \hat{f}|_{L \times L}$. **Claim:** $\sum_{a_1, a_2 \in H} \hat{f}(a_1, a_2) = \mu$.

My multivariate Sumcheck

Protocol

1. P computes $\hat{f}(X, Y)$.

My multivariate Sumcheck

Protocol

1. P computes $\hat{f}(X, Y)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.

My multivariate Sumcheck

Protocol

1. P computes $\hat{f}(X, Y)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.
3. P computes $\hat{g}$, $\hat{q}_1$, $\hat{q}_2$ such that

$$\widehat{f}(X, Y) = \widehat{g}(X, Y) + Z_H(X)\widehat{q}_1(X, Y) + Z_H(Y)\widehat{q}_2(X, Y), \deg_{X,Y} \widehat{g} < |H|.$$

My multivariate Sumcheck

Protocol

1. P computes $\hat{f}(X, Y)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.
3. P computes $\hat{g}$, $\hat{q}_1$, $\hat{q}_2$ such that

$$\widehat{f}(X, Y) = \widehat{g}(X, Y) + Z_H(X)\widehat{q}_1(X, Y) + Z_H(Y)\widehat{q}_2(X, Y), \deg_{X,Y} \widehat{g} < |H|.$$

4. P computes $\widehat{g}_1$, $\widehat{g}_2$, and $\beta \in \mathbb{F}_q$ such that:

$$\widehat{g}(X, Y) = \widehat{g}_1(X, Y) + Y^{|H|-1}\widehat{g}_2(X, Y) + \beta X^{|H|-1} Y^{|H|-1}.$$

My multivariate Sumcheck

Protocol

1. P computes $\hat{f}(X, Y)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.
3. P computes $\hat{g}$, $\hat{q}_1$, $\hat{q}_2$ such that

$$\widehat{f}(X, Y) = \widehat{g}(X, Y) + Z_H(X)\widehat{q}_1(X, Y) + Z_H(Y)\widehat{q}_2(X, Y), \deg_{X,Y} \widehat{g} < |H|.$$

4. P computes $\widehat{g}_1$, $\widehat{g}_2$, and $\beta \in \mathbb{F}_q$ such that:

$$\widehat{g}(X, Y) = \widehat{g}_1(X, Y) + Y^{|H|-1}\widehat{g}_2(X, Y) + \beta X^{|H|-1}Y^{|H|-1}.$$

5. P gives oracle access to V to $g_2 := \hat{g}_2|_{L \times L}$, $q_1 := \hat{q}_1|_{L \times L}$ and $q_2 := \hat{q}_2|_{L \times L}$.

My multivariate Sumcheck

Protocol

1. P computes $\hat{f}(X, Y)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.
3. P computes $\hat{g}, \hat{q}_1, \hat{q}_2$ such that
$$\widehat{f}(X, Y) = \widehat{g}(X, Y) + Z_H(X)\widehat{q}_1(X, Y) + Z_H(Y)\widehat{q}_2(X, Y), \deg_{X,Y} \widehat{g} < |H|.$$
4. P computes $\widehat{g}_1, \widehat{g}_2$, and $\beta \in \mathbb{F}_q$ such that:
$$\widehat{g}(X, Y) = \widehat{g}_1(X, Y) + Y^{|H|-1}\widehat{g}_2(X, Y) + \beta X^{|H|-1}Y^{|H|-1}.$$
5. P gives oracle access to V to $g_2 := \hat{g}_2|_{L \times L}$, $q_1 := \hat{q}_1|_{L \times L}$ and $q_2 := \hat{q}_2|_{L \times L}$.
6. V computes $\zeta = \left(\sum_{a \in H} a^{|H|-1}\right)^2$ and accepts if and only if
$$p \in RS[L, |H|] \otimes RS[L, |H| - 1]$$
where
$$\widehat{p} := \zeta(\widehat{f} - Y^{|H|-1}\widehat{g}_2 - \mu X^{|H|-1}Y^{|H|-1} - Z_H(X)\widehat{q}_1 - Z_H(Y)\widehat{q}_2).$$

My multivariate Sumcheck

Protocol

1. P computes $\hat{f}(X, Y)$.
2. P and V compute $Z_H(X) = \prod_{a \in H}(X - a)$.
3. P computes $\hat{g}, \hat{q}_1, \hat{q}_2$ such that

$$\widehat{f}(X, Y) = \widehat{g}(X, Y) + Z_H(X)\widehat{q}_1(X, Y) + Z_H(Y)\widehat{q}_2(X, Y), \deg_{X,Y} \widehat{g} < |H|.$$

4. P computes $\widehat{g}_1, \widehat{g}_2$, and $\beta \in \mathbb{F}_q$ such that:

$$\widehat{g}(X, Y) = \widehat{g}_1(X, Y) + Y^{|H|-1}\widehat{g}_2(X, Y) + \beta X^{|H|-1}Y^{|H|-1}.$$

5. P gives oracle access to V to $g_2 := \hat{g}_2|_{L \times L}$, $q_1 := \hat{q}_1|_{L \times L}$ and $q_2 := \hat{q}_2|_{L \times L}$.
6. V computes $\zeta = \left(\sum_{a \in H} a^{|H|-1}\right)^2$ and accepts if and only if

$$p \in RS[L, |H|] \otimes RS[L, |H| - 1]$$

where

$$\widehat{p} := \zeta(\widehat{f} - Y^{|H|-1}\widehat{g}_2 - \mu X^{|H|-1}Y^{|H|-1} - Z_H(X)\widehat{q}_1 - Z_H(Y)\widehat{q}_2).$$

7. V and P also runs low-degree tests to check the degrees of $\hat{g}_2(X, Y), \hat{q}_1(X, Y)$ and $\hat{q}_2(X, Y)$.

Our multivariate Sumcheck

Performance

- *Prover time:*
  - one 2*DIFFT* to get $\widehat{f}(X, Y)$.
  - one "divide-and-conquer" algorithms to get$Z_H(X)$ and $Z_H(Y) \rightarrow O(\log|H|)$
  - four polynomial divisions to compute $\widehat{g}_2(X, Y)$, β, $\widehat{q}_1(X, Y)$, $\widehat{q}_2(X, Y) \rightarrow O(M(d) \times d)$.
  - three 2*DFFT* to evaluate those polynomials over L^2.
  - four 2*DFRI*: $O(|L|^2)$.

  so Prover time in $O(\log|H| + M(d)d) + 4FFT(\mathbb{F}, L^2) + O(|L|^2)$.
- *Verifier time:* $O(\log^2|H|) + O(\log|H|)$, related to the 2*DFRI*.
- *Query complexity:* $O(4\log|H|)$, related to the 2*DFRI*.
- *Proof length:* $O(|L|^2)$.

Our multivariate Sumcheck

Performance

- *Prover time:* $O(\log|H| + M(d)d) + 4FFT(\mathbb{F}, L^2) + O(|L|^2)$.
- *Verifier time:* $O(\log^2|H|) + O(\log|H|)$, related to the 2*DFRI*.
- *Query complexity:* $O(4\log|H|)$, related to the 2*DFRI*.
- *Proof length:* $O(|L|^2)$.

If we have n variables

- *Prover time:* $O(n)FFT(\mathbb{F}, L^n) + O(\log|H|) + O(nM(d)d^{n-1}) + O(|L|^n)$.
- *Verifier time:* $O(n\log^2|H|) + O(n\log|H|)$.
- *Query complexity:* $O(n\log|H|)$.
- *Proof length:* $O(n|L|^n)$.

Comparison

	Sumcheck from [BCGRS17]	Our multivariate Sumcheck
Prover time	$\|L\|^n + npoly(\log\|\mathbb{F}\|) + n\tilde{O}(\|L\|^2 + \|H\|) + n\|L\|^n$	$O(n)FFT(\mathbb{F}, L^n) + O(\log\|H\|) + O(nM(d)d^{n-1}) + O(\|L\|^n)$
Verifier time	$poly(n + \|L\|) + npoly(\log\|\mathbb{F}\| + \log(\|L\|^2 + \|H\|)) + O(n)$	$O(n\log^2\|H\|) + O(n\log\|H\|)$
Proof length	$O(\|L\|^n\log(q) + n\tilde{O}(\|L\|^2 + \|H\|))$	$O(n\|L\|^n)$
Query complexity	$O(n)$	$O(n\log\|H\|)$

Conclusion

The univariate Sumcheck is well known and used, and it's efficiency is mostly due to the FRI protocol.

Since Sarah, Jade and Daniel made a multivariate version of the FRI, we made a multivariate version of the sumcheck that uses the FRI.

- it should have better performance in practice. Sumcheck from [3] Our multivariate Sumcheck
- it could be used within specific arithmetization with multivariate polynomials.

Conclusion

The univariate Sumcheck is well known and used, and it's efficiency is mostly due to the FRI protocol.

Since Sarah, Jade and Daniel made a multivariate version of the FRI, we made a multivariate version of the sumcheck that uses the FRI.

- it should have better performance in practice. Sumcheck from [3] Our multivariate Sumcheck
- it could be used within specific arithmetization with multivariate polynomials.

Thank you for listening!

[1] E. Ben-Sasson et al. “Fast Reed-Solomon Interactive Oracle Proofs of Proximity”. In: *Electron. Colloquium Comput. Complex.* 2017.

[2] Eli Ben-Sasson et al. *Aurora: Transparent Succinct Arguments for R1CS*. Cryptology ePrint Archive, Report 2018/828. `https://eprint.iacr.org/2018/828`. 2018.

[3] Eli Ben-Sasson et al. *Interactive Oracle Proofs with Constant Rate and Query Complexity*. Cryptology ePrint Archive, Report 2016/324. `https://ia.cr/2016/324`. 2016.

[4] Carsten Lund et al. “Algebraic Methods for Interactive Proof Systems”. In: vol. 39(4). Nov. 1990, 2–10 vol.1. ISBN: 0-8186-2082-X. DOI: `10.1109/FSCS.1990.89518`.