CS251 Fall 2025

(cs251.stanford.edu)

Building a SNARK

Dan Boneh

Recap: zk-SNARK applications

Private Tx on a public blockchain: Aztec, Aleo, Inco, …

Compliance:

- Proving that a private Tx are in compliance with banking laws
- Proving solvency in zero-knowledge

Scalability: zk-Rollup (next week)

Bridging between blockchains: zk-Bridge

Review: a preprocessing SNARK

Public arithmetic circuit: $C(\boldsymbol{x}, \boldsymbol{w}) \to \mathbb{F}$

public statement in $\mathbb{F}^n$ secret witness in $\mathbb{F}^m$

Preprocessing (setup): $\mathbf{S}(C) \to$ public parameters $(\boldsymbol{pp}, \boldsymbol{vp})$

$\boldsymbol{pp}, \boldsymbol{x}, \boldsymbol{w}$

$\boldsymbol{vp}, \boldsymbol{x}$

Prover

proof $\boldsymbol{\pi}$ that $C(x, w) = 0$

$\boldsymbol{\pi}$

Verifier

accept or reject

short:	$\mathrm{len}(\pi)$	$= O_\lambda(\, \mathbf{polylog}(\lvert \boldsymbol{C} \rvert))$
fast verify:	$\mathrm{time}(\mathrm{V})$	$= O_\lambda(\lvert x \rvert,\ \mathbf{polylog}(\lvert \boldsymbol{C} \rvert))$

SNARK: requirements (informal)

Prover P(pp, $\boldsymbol{x}$, $\boldsymbol{w}$) Verifier V (vp, $\boldsymbol{x}$, $\boldsymbol{\pi}$)

proof π → accept or reject

Complete: $\forall x, w$: $C(\boldsymbol{x}, \boldsymbol{w}) = 0 \Rightarrow$ Pr[V(vp, x, P(pp, $\boldsymbol{x}$, $\boldsymbol{w}$)) = accept] = 1

Adaptively **knowledge sound**: V accepts $\Rightarrow$ P "knows" $\boldsymbol{w}$ s.t. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$

(an extractor E can extract a valid $\boldsymbol{w}$ from P)

Optional: **Zero knowledge**: $(C, pp, vp, \boldsymbol{x}, \pi)$ "reveal nothing new" about $\boldsymbol{w}$

(witness exists $\Rightarrow$ can simulate the proof)

A simple PCP-based SNARK

[Kilian'92, Micali'94]

A simple construction: PCP-based SNARK

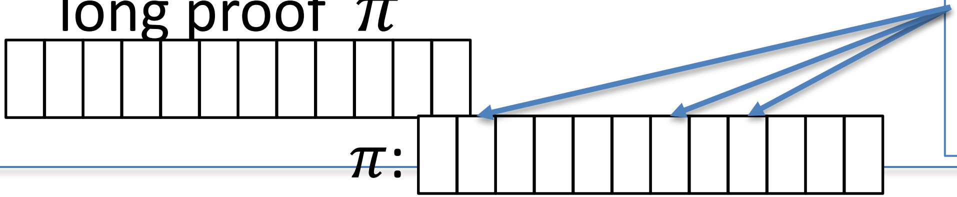
The PCP theorem: (1992) Let $C(x, w)$ be an arithmetic circuit.

there is a proof system that for every x proves $\exists w: C(x, w) = 0$ as follows:

Prover P(pp, $\boldsymbol{x}$, $\boldsymbol{w}$)

Verifier V(vp, $\boldsymbol{x}$)

long proof π



π:

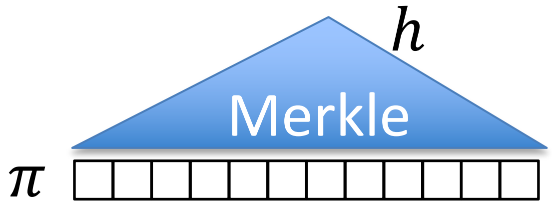
read only $O(\lambda)$ bits of π,
output accept or reject

V always accepts valid proof. If no w, then V rejects with high prob.

size of proof π is $poly(|C|)$. (not succinct)

Convert a PCP to a SNARK: making proof short

Prover P(*pp*, $\boldsymbol{x}$, $\boldsymbol{w}$) | Verifier V(*vp*, $\boldsymbol{x}$)



commit to π

Merkle root h → 1 hash

← open k positions of π $(k = O(\lambda))$

→ k opening and Merkle proofs — $O(k \log |C|)$ hashes

output accept or reject

Verifier sees $O(\lambda \log |C|)$ data $\Rightarrow$ succinct proof.

Problem: **interactive**

Making the proof non-interactive

The **Fiat-Shamir transform**:

- public-coin interactive protocol $\Rightarrow$ non-interactive protocol

  public coin: all verifier randomness is public (no secrets)

Prover P(pp, $\boldsymbol{x}$, $\boldsymbol{w}$)		Verifier V(vp, $\boldsymbol{x}$)
	$\xrightarrow{\text{msg1}}$	
	$\xleftarrow{\text{r}}$	choose random bits r
	$\xrightarrow{\text{msg2}}$	
		accept or reject

Making the proof non-interactive

Fiat-Shamir transform: $H: M \to R$ a cryptographic hash function

- idea: prover generates random bits on its own (!)

Prover P(pp, $\boldsymbol{x}$, $\boldsymbol{w}$)

generate msg1

r $\leftarrow$ H($\boldsymbol{x}$, msg1)

generate msg2

π = (msg1, msg2)

$|\pi| = O(\lambda \log |C|)$

Verifier V(vp, $\boldsymbol{x}$)

r $\leftarrow$ H($\boldsymbol{x}$, msg1)

accept or reject

Fiat-Shamir: certain secure interactive protocols $\Longrightarrow$ non-interactive

Are we done?

Simple transparent SNARK from the PCP theorem

- Use Fiat-Shamir transform to make non-interactive
- We will apply Fiat-Shamir in many other settings

The bad news: an impractical SNARK --- Prover time too high

Better SNARKs: Goal: Time(Prover) = $\tilde{O}(|C|)$

Building an efficient SNARK

General paradigm: two steps

(1) A polynomial commitment scheme

(cryptographic object)

(2) A polynomial interactive oracle proof (Poly-IOP)

(info. theoretic object)

SNARK for general circuits

Let’s explain each concept …

(1) Polynomial commitment scheme (PCS)

Notation: $\mathbb{F}_p^{(\leq d)}[X]$ is all polynomials in $\mathbb{F}_p[X]$ of degree ≤ d.

Prover commits to a polynomial $f(X)$ in $\mathbb{F}_p^{(\leq d)}[X]$ (univariate)

- ***eval***: for public $u, v \in \mathbb{F}_p$, prover can convince the verifier that committed poly satisfies

$$\boxed{f(u) = v \quad \text{and} \quad \deg(f) \leq d.}$$

verifier has (d, com_f, u, v)

$\boxed{f}$

- Eval proof size and verifier time should be $O_\lambda(\log \boldsymbol{d})$

(1) Polynomial commitment scheme (PCS)

- *setup*$(d) \to pp$, public parameters for polynomials of degree $\le d$
- *commit*$(pp, \mathrm{f}, r) \to \boldsymbol{com_f}$ commitment to $\mathrm{f} \in \mathbb{F}_p^{(\le d)}[X]$
- *eval*: goal: for a given $\boldsymbol{com_f}$ and x, y $\in \mathbb{F}_p$, prove that f(x) = y.

Formally: *eval* = (S, P, V) is a SNARK for:

statement $st = (pp, \boldsymbol{com_f}, \mathrm{x}, \mathrm{y})$ with witness = w = (f, r)

where $C(st, w) = 0$ iff

$$\left[\ \mathrm{f(x) = y} \quad \textit{and} \quad \mathrm{f} \in \mathbb{F}_p^{(\le d)}[X] \quad \textit{and} \quad \textit{commit}(pp, \mathrm{f}, r) = \boldsymbol{com}_{\mathrm{f}}\ \right]$$

(1) Polynomial commitment scheme (PCS)

Properties:

- Binding: cannot produce two valid openings (f_1, r_1), (f_2, r_2) for $\boldsymbol{com_f}$.
- eval is knowledge sound (can extract (f, r) from a successful prover)
- optional:
  - commitment is hiding
  - eval is zero knowledge

Note: poly. commitments have many applications beyond SNARKs

Constructing a PCS

Not today …   (see readings or CS355)

Properties of a famous PCS (called KZG) :

- trusted setup: secret randomness in setup. $\boxed{|pp| = O_\lambda(d)}$
- ***com***$_f$: constant size (one group element)
- eval proof size: constant size (one group element)
- eval verify time: constant time.   Prover time: $\boxed{O_\lambda(d)}$

General paradigm: two steps

(1) A polynomial commitment scheme

(cryptographic object)

(2) A polynomial interactive oracle proof (Poly-IOP)

(info. theoretic object)

SNARK for general circuits

What is a Poly-IOP?

Polynomial IOPs (PIOPs)

Setup(C) → public parameters pp and $\left(vp, \boxed{g, w \in \mathbb{F}_p^{(\leq d)}[X]}\right)$

Prover $P(pp, \mathbf{x}, \mathbf{w})$

Verifier $V^{g,w}(vp, \mathbf{x})$

oracle $\boxed{f_0 \in \mathbb{F}_p^{(\leq d)}[X]}$ →

$r_1 \leftarrow \mathbb{F}_p$

← r_1

⋮

← r_{t-1}

$r_{t-1} \leftarrow \mathbb{F}_p$

oracle $\boxed{f_t \in \mathbb{F}_p^{(\leq d)}[X]}$ →

Verifier is assured that all oracles are in $\mathbb{F}_p^{(\leq d)}[X]$

$$\mathrm{verify}^{g,w,f_0,\ldots,f_t}(\mathbf{x}, r_1, , \ldots, r_{t-1}) \to 0/1$$

Polynomial IOPs (PIOPs)

Def: Let $C(\mathbb{x}, \mathbb{w})$ a circuit. A PIOP $(\boldsymbol{S}, \boldsymbol{P}, \boldsymbol{V})$ for C

is **complete** if for all $(\mathbb{x}, \mathbb{w})$, $C(\mathbb{x}, \mathbb{w}) = 0$, when $\boldsymbol{V}$ interacts with $\boldsymbol{P}$

$$\Pr[\ \boldsymbol{V}^{g, w, f_0, \ldots, f_t}(\mathbb{x}, r_1, \ldots, r_k) = \text{yes}\ ] = 1$$

is **sound** if for all $\boldsymbol{P}^*$ and $\mathbb{x} \notin L(C) \coloneqq \{\mathbb{x} \mid \exists \mathbb{w}\colon C(\mathbb{x}, \mathbb{w}) = 0\}$ we have that

$$\Pr[\ \boldsymbol{V}^{g, w, f_0, \ldots, f_t}(\mathbb{x}, r_1, \ldots, r_k) = \text{yes}\ ] < err \qquad (\approx 2^{-128})$$

is **succinct** if time$(\boldsymbol{V})$ is at most $\mathrm{polylog}(|C|)$ and $O(|\mathbb{x}|, \log(1/err))$

$\Rightarrow$ t is small and $\boldsymbol{V}$ makes few queries to its oracles

The paradigm: Poly-IOP + PCS ⇒ SNARK

Replace poly. oracles with
poly. commitments
and evaluation proofs

Fiat-Shamir
(to remove interaction)

Polynomial
interactive
oracle proof
(**Poly-IOP**)

Interactive
Proof (IP)

(zk)SNARK for
general circuits

BCS compiler [BCS'16]

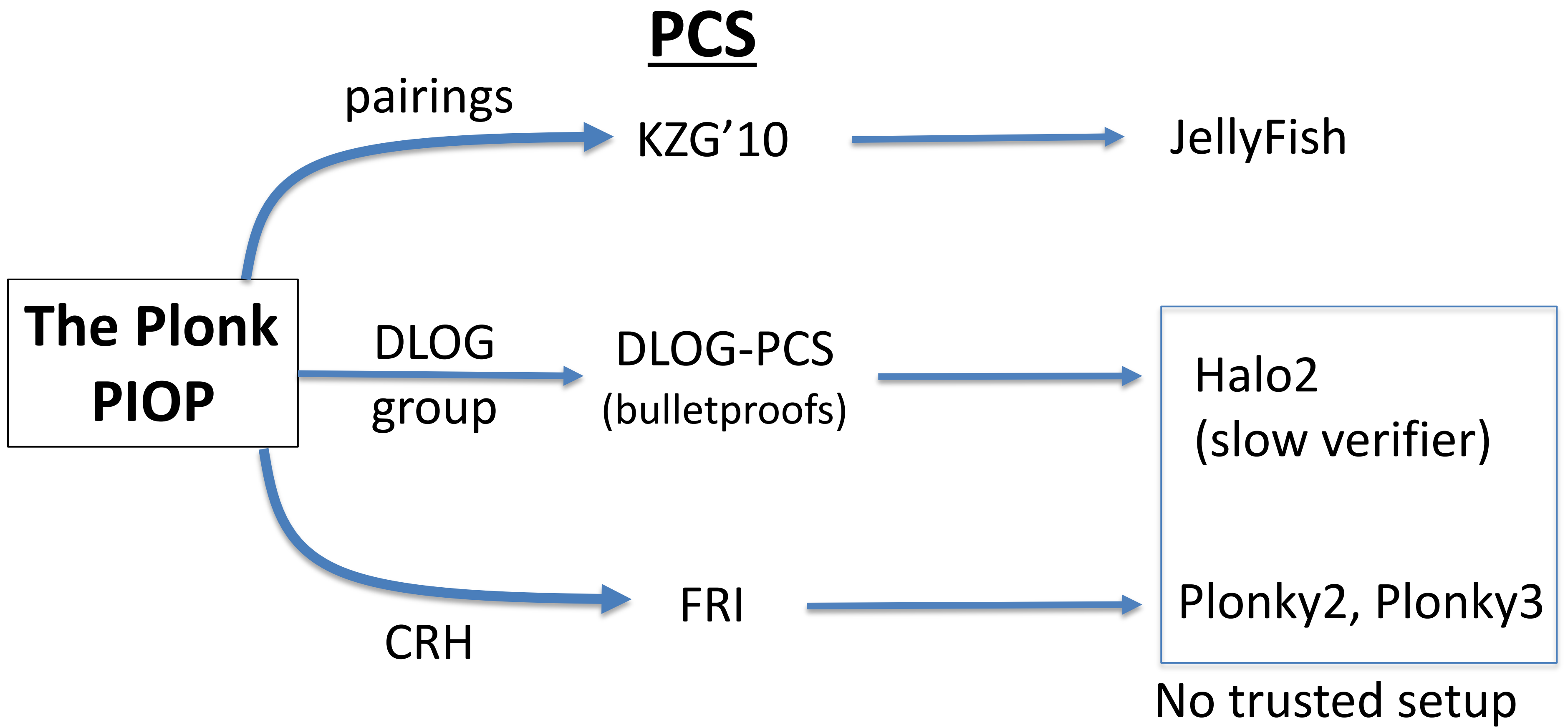
The Plonk poly-IOP (eprint/2019/953)

Gabizon – Williamson – Ciobotaru

Plonk PIOP + Polynomial Commitment $\Rightarrow$ SNARK

(and also a zk-SNARK)

Plonk Systems

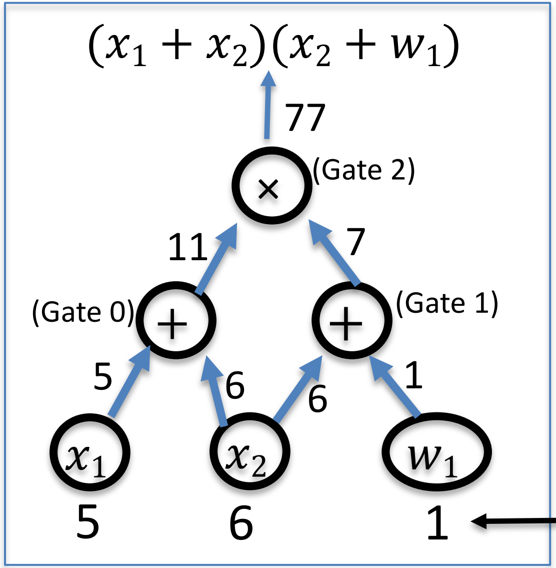


The PLONK PIOP

eprint/2019/953

PLONK: a poly-IOP for a general circuit $C(x, w)$

Step 1: compile circuit to a computation trace (arithmetization)



example input

The computation trace (arithmetization):

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	77

left inputs

right inputs

outputs

Encoding the trace as a polynomial

$|C| :=$ total # of gates in C , $|I| := |I_x| + |I_w| =$ # inputs to C

let $d := 3\,|C| + |I|$ (in example, $d = 12$) and $\Omega := \{\, 1, \omega, \omega^2, \ldots, \omega^{d-1} \,\}$

The plan:

prover interpolates a poly. $T \in \mathbb{F}_p^{(\leq d)}[X]$

that encodes the entire trace.

Let's see how …

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	$\boxed{77}$

Encoding the trace as a polynomial

The plan: Prover interpolates $T \in \mathbb{F}_p^{(\le d)}[X]$ such that

(1) **T encodes all inputs**: $T(\omega^{-j}) = \text{input } \#j$ for $j = 1, \ldots, |I|$

(2) **T encodes all wires**: $\forall\, l = 0, \ldots, |C| - 1$:

- $T(\omega^{3l})$: left input to gate $\#l$
- $T(\omega^{3l+1})$: right input to gate $\#l$
- $T(\omega^{3l+2})$: output of gate $\#l$

Plonk PIOP:
- send oracle for T
- prove T is valid (gates and wires)

Encoding the trace as a polynomial

In our example, Prover interpolates $T(X)$ such that:

inputs:	$T(\omega^{-1}) = 5,$	$T(\omega^{-2}) = 6,$	$T(\omega^{-3}) = 1,$
gate 0:	$T(\omega^{0}) = 5,$	$T(\omega^{1}) = 6,$	$T(\omega^{2}) = 11,$
gate 1:	$T(\omega^{3}) = 6,$	$T(\omega^{4}) = 1,$	$T(\omega^{5}) = 7,$
gate 2:	$T(\omega^{6}) = 11,$	$T(\omega^{7}) = 7,$	$T(\omega^{8}) = 77$

degree(T) = 11

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	$\boxed{77}$

Prover can use FFT to compute the coefficients of T in time $O(d \log d)$

Step 2: proving validity of T

Prover P($pp, \boldsymbol{x}, \mathbf{w}$)		Verifier V($vp, \boldsymbol{x}$)
build T(X) $\in \mathbb{F}_p^{(\le d)}$[X]	$\xrightarrow{\boxed{T}}$	

Prover needs to prove that T is a correct computation trace:

(1) T encodes the correct inputs,
(2) every gate is evaluated correctly,
(3) the wiring is implemented correctly,
(4) the output of last gate is 0

How? First, let's build some tools.

(wiring constraints)

inputs:	5 ,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	77

Towards the Plonk PIOP

Proving properties of committed univariate polynomials

Proving properties of committed polynomials

Prover P(f, g)

Verifier V($\boxed{f}$, $\boxed{g}$)

Goal: convince verifier that $f, g \in \mathbb{F}_p^{(\leq d)}[X]$ satisfy some properties

Proof systems presented as a Poly-IOP:

r

$r \leftarrow \mathbb{F}_p$

$\boxed{q}$ (a commitment to some poly. q)

query $f(X), g(X), q(X)$ at some points in $\mathbb{F}_p$

accept or reject

An example: polynomial equality testing

Prover

$\boldsymbol{f}, \boldsymbol{g} \in \mathbb{F}_p^{(\leq d)}[X]$

Goal: convince verifier that $f = g$

query $f(X)$ and $g(X)$ at r

Verifier

$\boxed{f}$ $\boxed{g}$

$r \leftarrow \mathbb{F}_p$

learn $f(r), g(r)$

accept if:
$f(r) = g(r)$

Why is this sound?

Why is this sound?

A key fact: for non-zero $f \in \mathbb{F}_p^{(\leq d)}[X]$

$$\boxed{\text{for } r \leftarrow \mathbb{F}_p : \quad \Pr[\, f(r) = 0 \,] \leq d/p} \qquad (*)$$

$\Rightarrow$ suppose $p \approx 2^{256}$ and $d \leq 2^{40}$ then d/p is negligible

$\Rightarrow$ for $r \leftarrow \mathbb{F}_p$: if $f(r) = 0$ then f is identically zero w.h.p

$\Rightarrow$ a simple test if a committed poly. is the zero poly.

SZDL lemma: $(*)$ also holds for **multivariate** polynomials (where d is total degree of f)

Why is this sound?

Suppose $p \approx 2^{256}$ and $d \leq 2^{40}$ so that d/p is negligible

Let $f, g \in \mathbb{F}_p^{(\leq d)}[X]$.

For $r \leftarrow \mathbb{F}_p$, if $f(r) = g(r)$ then $f = g$ w.h.p

$f(r) - g(r) = 0 \quad \Rightarrow \quad f - g = 0$ w.h.p

$\Rightarrow$ a simple equality test for two committed polynomials

The polynomial equality testing protocol

Prover

Goal: convince verifier that $f = g$

Verifier

$\boldsymbol{f}, \boldsymbol{g} \in \mathbb{F}_p^{(\leq d)}[X]$

$\boxed{f}$ $\boxed{g}$

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

query $f(\mathrm{X})$ and $g(X)$ at $\mathrm{X} = r$

learn $f(r), \mathrm{g}(r)$

accept if:
$f(r) = \mathrm{g}(r)$

Lemma: complete and sound assuming d/p is negligible

The compiled proof system

Make non-interactive using Fiat-Shamir

Prover

$f, g \in \mathbb{F}_p^{(\leq d)}[X]$

$y \leftarrow f(r)$

$y' \leftarrow g(r)$

r

$y, \ \pi_f$

$y', \ \pi_g$

proof that $y = f(r)$

proof that $y' = g(r)$

Verifier

$com_f \, , com_g$

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

learn $f(r), g(r)$

accept if:

(i) $y = y'$ and

(ii) π_f, π_g are valid

Important proof gadgets for univariates

Let Ω be some subset of $\mathbb{F}_p$ of size k.

Let $f \in \mathbb{F}_p^{(\le d)}[X]$ $\quad (d \ge k)$ $\quad$ Verifier has $\boxed{f}$

Let us construct efficient Poly-IOPs for the following tasks:

Task 1 (**ZeroTest**): prove that f is identically zero on Ω

Task 2 (**SumCheck**): prove that $\sum_{a\in\Omega} f(a) = 0$

Task 3 (**ProdCheck**): prove that $\prod_{a\in\Omega} f(a) = 1$

The vanishing polynomial

Let Ω be some subset of $\mathbb{F}_p$ of size k.

Def: the **vanishing polynomial** of Ω is $Z_\Omega(X) := \prod_{a \in \Omega}(X - a)$

$\deg(Z_\Omega) = k$

Let $\omega \in \mathbb{F}_p$ be a primitive k-th root of unity (so that $\omega^k = 1$).

- if $\Omega = \{1, \omega, \omega^2, \ldots, \omega^{k-1}\} \subseteq \mathbb{F}_p$ then $Z_\Omega(X) = X^k - 1$

$\Rightarrow$ for $r \in \mathbb{F}_p$, evaluating $Z_\Omega(r)$ takes $2\log_2 k$ field operations

(1) ZeroTest on Ω ($\Omega = \{1, \omega, \omega^2, \ldots, \omega^{k-1}\}$)

Prover P(f)

Verifier V($\boxed{f}$)

$q(X) \leftarrow f(X)/Z_\Omega(X)$

$\boxed{q \in \mathbb{F}_p^{(\le d)}[X]}$

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

verifier evaluates $Z_\Omega(r)$ by itself

query $q(X)$ and $f(X)$ at r

learn $q(r)$, $f(r)$

accept if $f(r) \stackrel{?}{=} q(r) \cdot Z_\Omega(r)$

(implies that $f(X) = q(X) \cdot Z_\Omega(X)$ w.h.p)

Lemma: f is zero on Ω if and only if $f(X)$ is divisible by $Z_\Omega(X)$

Thm: this protocol is complete and sound, assuming d/p is negligible.

(1) ZeroTest on Ω $\quad$ ($\Omega = \{ 1, \omega, \omega^2, \ldots, \omega^{k\text{-}1} \}$)

Prover P(f) $\qquad\qquad$ Verifier V($\boxed{f}$)

$q(X) \leftarrow f(X)/Z_\Omega(X)$

$\boxed{q \in \mathbb{F}_p^{(\leq d)}[X]}$ →

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

verifier evaluates $Z_\Omega(r)$ by itself

query $q(X)$ and $f(X)$ at r ↔

learn $q(r)$, $f(r)$

accept if $f(r) \stackrel{?}{=} q(r) \cdot Z_\Omega(r)$

(implies that $f(X) = q(X) \cdot Z_\Omega(X)$ w.h.p)

Lemma: f is zero on Ω if and only if $f(X)$ is divisible by $Z_\Omega(X)$

Verifier time: O(log k) and two poly queries (but can be batched)

Prover time: dominated by time to compute $q(X)$ [and commit to $q(X)$]

(3) Product check on Ω : $\prod_{a\in\Omega} f(a) = 1$

Set $t \in \mathbb{F}_p^{(\leq k)}[X]$ to be the degree-d polynomial:

$$t(1) = f(1), \qquad t(\omega^s) = \prod_{i=0}^{s} f(\omega^i) \quad \text{for} \ s = 1, \dots, k-1$$

Then $t(\omega^{k-1}) = \prod_{a\in\Omega} f(a)$

and $t(\omega \cdot x) = t(x) \cdot f(\omega \cdot x)$ for all $x \in \Omega$ (including $x = \omega^{k-1}$)

Lemma: if (1) $t(\omega^{k-1}) = 1$ and

(2) $t_1(x) \coloneqq t(\omega \cdot x) - t(x) \cdot f(\omega \cdot x) = 0 \quad \forall x \in \Omega$

then $\prod_{a\in\Omega} f(a) = 1$

(3) Product check on Ω (unoptimized)

Prover P(f)

Verifier V($\boxed{f}$)

construct $t(X) \in \mathbb{F}_p^{(\leq k)}$, $t_1(X) := t(\omega \cdot X) - t(X) \cdot f(\omega \cdot X)$

and $q(X) := t_1(X)/(X^k - 1) \in \mathbb{F}_p^{(\leq d)}$

$\boxed{q\ ,\ \ t}$ $\longrightarrow$

$r \leftarrow \mathbb{F}_p$

query $t(X)$ at ω^{k-1} , r, ωr $\longrightarrow$

query $q(X)$ at r , and $f(X)$ at ωr $\longrightarrow$

learn $t(\omega^{k-1})$, $t(r)$, $t(\omega r)$, $q(r)$, $f(\omega r)$

$t_1(\Omega) = 0 \quad \Leftarrow$

accept if $t(\omega^{k-1}) \stackrel{?}{=} 1$ and
$t(\omega r) - t(r) f(\omega r) \stackrel{?}{=} q(r) \cdot (r^k - 1)$

Complete and sound, assuming $\deg(t_1)/p = (k + d)/p$ is negligible.

Same works for rational functions: $\prod_{a\in\Omega}(f/g)(a) = 1$

Prover P(f, g)

Verifier V($\boxed{f}$, $\boxed{g}$)

Set $t \in \mathbb{F}_p^{(\le k)}[X]$ to be the degree-k polynomial:

$$t(1) = f(1)/g(1), \qquad t(\omega^s) = \prod_{i=0}^{s} f(\omega^i)/g(\omega^i) \quad \text{for } s = 1, \dots, k-1$$

Lemma: if (i) $t(\omega^{k-1}) = 1$ and
(ii) $t(\omega \cdot x) \cdot g(\omega \cdot x) = t(x) \cdot f(\omega \cdot x)$ for all $x \in \Omega$

then $\prod_{a\in\Omega} f(a)/g(a) = 1$

(4) Another useful gadget: permutation check

Let f, g polynomials in $\mathbb{F}_p^{(\le d)}[X]$. Verifier has $\boxed{f}$, $\boxed{g}$.

Prover wants to prove that $\left(f(1), f(\omega), f(\omega^2), \dots, f(\omega^{k-1})\right) \in \mathbb{F}_p^k$

is a permutation of $\left(g(1), g(\omega), g(\omega^2), \dots, g(\omega^{k-1})\right) \in \mathbb{F}_p^k$

$\Rightarrow$ Proves that $g(\Omega)$ is the same as $f(\Omega)$, just permuted

(4) Another useful gadget: permutation check

Prover P(f, g) Verifier V($\boxed{f}$, $\boxed{g}$)

Let $\hat{f}(X) = \prod_{a\in\Omega}(X - f(a))$ and $\hat{g}(X) = \prod_{a\in\Omega}(X - g(a))$

Then: $\hat{f}(X) = \hat{g}(X) \iff g(\Omega)$ is a permutation of $f(\Omega)$

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

prove that $\hat{f}(r) = \hat{g}(r)$

prod-check: $\frac{\hat{f}(r)}{\hat{g}(r)} = \prod_{a\in\Omega}\left(\frac{r - f(a)}{r - g(a)}\right) = 1$

$\longleftrightarrow$ implies $\hat{f}(X) = \hat{g}(X)$ w.h.p

accept or reject

[Lipton's trick, 1989]

(4') Permutation check on pairs

Let f_1, f_2, g_1, g_2 be polynomials in $\mathbb{F}_p^{(\leq d)}[X]$.

Prover wants to prove that the k pair

$$\Big(\big(f_1(1), f_2(1)\big), \ldots, \big(f_1(\omega^{k-1}), f_2(\omega^{k-1})\big)\Big) \in (\mathbb{F}_p^2)^k$$

are a permutation of

$$\Big(\big(g_1(1), g_2(1)\big), \ldots, \underbrace{\big(g_1(\omega^{k-1}), g_2(\omega^{k-1})\big)}_{\text{one pair}}\Big) \in (\mathbb{F}_p^2)^k$$

(4’) Permutation check on pairs

Define: $\hat{f}(X,Y) \coloneqq \prod_{a\in\Omega} \left(X - Y\cdot f_1(a) - f_2(a)\right)$ and

$\hat{g}(X,Y) \coloneqq \prod_{a\in\Omega} \left(X - Y\cdot g_1(a) - g_2(a)\right)$

Lemma: $\hat{f}(X,Y) = \hat{g}(X,Y)$ if and only if

$\left(f_1(a), f_2(a)\right)_{a\in\Omega}$ is a permutation of $\left(g_1(a), g_2(a)\right)_{a\in\Omega}$

To prove, use the fact that $\mathbb{F}_p[X,Y]$ is a unique factorization domain

Now: $\hat{f}(X,Y) = \hat{g}(X,Y)$ can be checked using a product check (using $X, Y \leftarrow \mathbb{F}_p$)

The protocol

Prover P(f_1, f_2, g_1, g_2)

Verifier V($\boxed{f_1, f_2}$, $\boxed{g_1, g_2}$)

r, s

$r, s \leftarrow \mathbb{F}_p$

prove that $\hat{f}(r, s) = \hat{g}(r, s)$:

ProdCheck: $$\prod_{a \in \Omega} \left(\frac{r - s \cdot f_1(a) - f_2(a)}{r - s \cdot g_1(a) - g_2(a)} \right) = 1$$

by Schwartz-Zippel

implies $\hat{f}(X, Y) = \hat{g}(X, Y)$ w.h.p

accept or reject

Complete and sound, assuming $(k + d)/p$ is negligible.

(5) final gadget: prescribed permutation check

$W: \Omega \to \Omega$ is a **permutation of** Ω if $\forall i \in [k]$: $W(\omega^i) = \omega^j$ is a bijection

example $(k = 3)$: $W(\omega^0) = \omega^2$, $W(\omega^1) = \omega^0$, $W(\omega^2) = \omega^1$

Let f, g polynomials in $\mathbb{F}_p^{(\leq d)}[X]$. Verifier has $\boxed{f}$, $\boxed{g}$, $\boxed{W}$.

<u>Goal</u>: prover wants to prove that $f(y) = g(W(y))$ for all $y \in \Omega$

$\Rightarrow$ Proves that $g(\Omega)$ is the same as $f(\Omega)$, permuted by the prescribed W

Prescribed permutation check

How? Use a <u>zero-test</u> to prove $\boxed{f(y) - g\big(W(y)\big) = 0 \quad \text{on } \Omega}$

<u>**The problem**</u>: the polynomial $f(y) - g\big(W(y)\big)$ has degree kd

$\Rightarrow$ prover would need to manipulate polynomials of degree kd

$\Rightarrow$ quadratic time prover !! (goal: linear time prover)

<u>**Goal**</u>: reduce this to a perm. check on pairs for degree-d poly (not kd)

Prescribed permutation check

Observation:

if $\big(W(a), f(a)\big)_{a\in\Omega}$ is a permutation of $\big(a, g(a)\big)_{a\in\Omega}$

then $f(y) = g(W(y))$ for all $y \in \Omega$

Proof by example: $W(\omega^0) = \omega^2$, $W(\omega^1) = \omega^0$, $W(\omega^2) = \omega^1$

Right tuple: $(\omega^0, g(\omega^0))$, $(\omega^1, g(\omega^1))$, $(\omega^2, g(\omega^2))$

Left tuple: $(\omega^2, f(\omega^0))$, $(\omega^0, f(\omega^1))$, $(\omega^1, f(\omega^2))$

So: permutation check on pairs $\Rightarrow$ prescribed permutation check

Summary of proof gadgets

prescribed permutation check

permutation check on pairs

product check, sum check

zero test on Ω

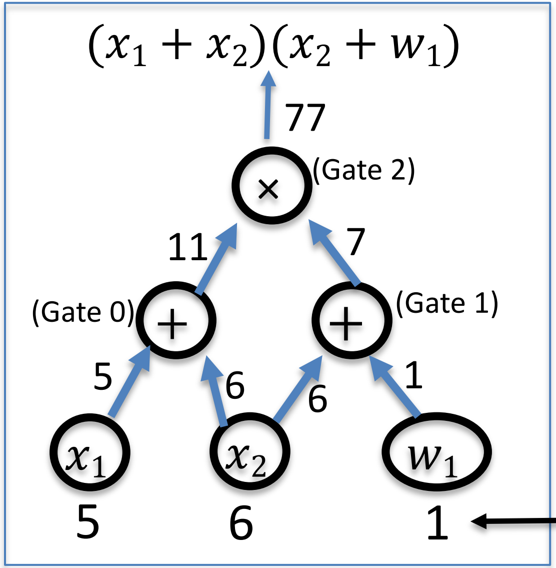
polynomial equality testing

The PLONK Poly-IOP
for general circuits

eprint/2019/953

PLONK: a poly-IOP for a general circuit $C(x, w)$

Step 1: compile circuit to a computation trace (gate fan-in = 2)



example input

The computation trace (arithmetization):

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	77

left inputs

right inputs

outputs

Encoding the trace as a polynomial

$|C| :=$ total # of gates in C , $\quad |I| := |I_x| + |I_w| =$ # inputs to C

let $d := 3\,|C| + |I|$ (in example, $d = 12$) and $\Omega := \{\, 1, \omega, \omega^2, \dots, \omega^{d-1} \,\}$

The plan:

prover interpolates a poly. $T \in \mathbb{F}_p^{(\leq d)}[\mathrm{X}]$

that encodes the entire trace.

Let's see how …

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	$\boxed{77}$

Encoding the trace as a polynomial

The plan: Prover interpolates $T \in \mathbb{F}_p^{(\leq d)}[X]$ such that

(1) ***T* encodes all inputs**: $T(\omega^{-j}) = \text{input } \#j$ for $j = 1, \ldots, |I|$

(2) ***T* encodes all wires**: $\forall\, l = 0, \ldots, |C| - 1$:

- $T(\omega^{3l})$: left input to gate $\#l$
- $T(\omega^{3l+1})$: right input to gate $\#l$
- $T(\omega^{3l+2})$: output of gate $\#l$

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	$\boxed{77}$

Encoding the trace as a polynomial

In our example, Prover interpolates $T(X)$ such that:

inputs:	$T(\omega^{-1}) = 5,$	$T(\omega^{-2}) = 6,$	$T(\omega^{-3}) = 1,$
gate 0:	$T(\omega^{0}) = 5,$	$T(\omega^{1}) = 6,$	$T(\omega^{2}) = 11,$
gate 1:	$T(\omega^{3}) = 6,$	$T(\omega^{4}) = 1,$	$T(\omega^{5}) = 7,$
gate 2:	$T(\omega^{6}) = 11,$	$T(\omega^{7}) = 7,$	$T(\omega^{8}) = 77$

degree(T) = 11

inputs:	5,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	$\boxed{77}$

Prover can use FFT to compute the coefficients of T in time $O(d \log d)$

Step 2: proving validity of T

Prover $\mathrm{P}(S_p, \boldsymbol{x}, \mathbf{w})$

Verifier $\mathrm{V}(S_v, \boldsymbol{x})$

build $T(X) \in \mathbb{F}_p^{(\le d)}[\mathrm{X}]$ $\xrightarrow{\boxed{T}}$

Prover needs to prove that T is a correct computation trace:

(1) T encodes the correct inputs,
(2) every gate is evaluated correctly,
(3) the wiring is implemented correctly,
(4) the output of last gate is 0

Proving (4) is easy: prove $T(\omega^{3|C|-1}) = 0$

(wiring constraints)

inputs:	5 ,	6,	1
Gate 0:	5 ,	6 ,	11
Gate 1:	6 ,	1 ,	7
Gate 2:	11,	7,	77

Proving (1): T encodes the correct inputs

Both prover and verifier interpolate a polynomial $v(X) \in \mathbb{F}_p^{(\leq|I_x|)}[X]$ that encodes the x-inputs to the circuit:

$$\text{for } j = 1, \ldots, |I_x|: \qquad v(\omega^{-j}) = \text{input \#j}$$

In our example: $v(\omega^{-1}) = 5, \quad v(\omega^{-2}) = 6$. (v is linear)

constructing $v(X)$ takes time proportional to the size of input x

$\Rightarrow$ verifier has time do this

Proving (1): T encodes the correct inputs

Both <u>prover</u> and <u>verifier</u> interpolate a polynomial $v(X) \in \mathbb{F}_p^{(\leq |I_x|)}[X]$ that encodes the x-inputs to the circuit:

$$\text{for } j = 1, \ldots, |I_x|: \qquad v(\omega^{-j}) = \text{input \#j}$$

Let $\Omega_{\text{inp}} := \{\, \omega^{-1}, \omega^{-2}, \ldots, \omega^{-|I_x|} \,\} \subseteq \Omega$ (points encoding the input)

Prover proves (1) by using a ZeroTest on Ω_{inp} to prove that

$$\boxed{T(y) - v(y) = 0 \qquad \forall\, y \in \Omega_{\text{inp}}}$$

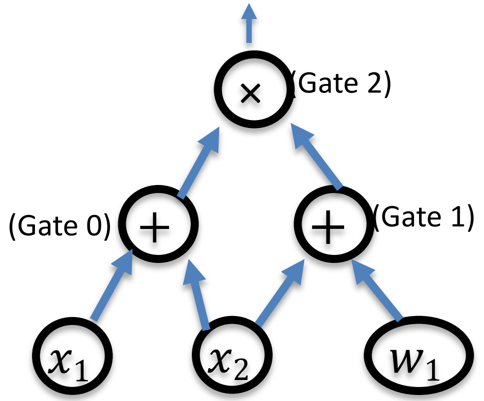
Proving (2): every gate is evaluated correctly

Idea: encode gate types using a *selector* polynomial S(X)

define S(X) $\in \mathbb{F}_p^{(\leq d)}[X]$ such that $\forall\, l = 0, \ldots, |C| - 1$:

S(ω^{3l}) = 1 if gate #l is an addition gate

S(ω^{3l}) = 0 if gate #l is a multiplication gate



inputs:	5,	6,	1	$S(X)$	
Gate 0 (ω^0):	5,	6,	11	1	(+)
Gate 1 (ω^3):	6,	1,	7	1	(+)
Gate 2 (ω^6):	11,	7,	77	0	(×)

Proving (2): every gate is evaluated correctly

Idea: encode gate types using a *selector* polynomial S(X)

define $S(X) \in \mathbb{F}_p^{(\leq d)}[X]$ such that $\forall\, l = 0, \ldots, |C| - 1$:

$S(\omega^{3l}) = 1$ if gate $\#l$ is an addition gate

$S(\omega^{3l}) = 0$ if gate $\#l$ is a multiplication gate

Then $\forall\, y \in \Omega_{\text{gates}} := \{\, 1, \omega^3, \omega^6, \omega^9, \ldots, \omega^{3(|C|-1)} \,\}$:

$$S(y)\cdot[T(y) + T(\omega y)] \quad + \quad (1 - S(y))\cdot T(y)\cdot T(\omega y) \quad = \quad T(\omega^2 y)$$

left input | right input | left input | right input | output

Proving (2): every gate is evaluated correctly

$$\text{Setup}(C) \;\rightarrow\; pp{:=}\text{S} \quad \text{and} \quad vp{:=}\left(\,\boxed{\text{S}}\,\right)$$

<u>Prover P($pp, \boldsymbol{x}, \mathbf{w}$)</u>

<u>Verifier V($vp, \boldsymbol{x}$)</u>

build $\text{T}(X) \in \mathbb{F}_p^{(\leq d)}[\text{X}]$ $\xrightarrow{\boxed{T}}$

Prover uses ZeroTest to prove that for all $\forall\, y \in \Omega_{gates}$:

$$\text{S}(\text{y})\cdot[\text{T}(\text{y}) + \text{T}(\omega\text{y})] \;+\; (1 - \text{S}(\text{y}))\cdot\text{T}(\text{y})\cdot\text{T}(\omega\text{y}) \;-\; \text{T}(\omega^2\text{y}) \;=\; 0$$

Proving (3): T respects the wires of C

Copy constraints:

$$\begin{cases} T(\omega^{-2}) = T(\omega^{1}) = T(\omega^{3}) \\ T(\omega^{-1}) = T(\omega^{0}) \\ T(\omega^{2}) = T(\omega^{6}) \\ T(\omega^{-3}) = T(\omega^{4}) \end{cases}$$

<u>example:</u> $x_1{=}5,\ x_2{=}6\ ,\ w_1{=}1$

	$\omega^{-1},\ \omega^{-2},\ \omega^{-3}$:	**5,**	**6,**	**1**
0:	$\omega^{0},\ \omega^{1},\ \omega^{2}$:	**5,**	**6,**	**11**
1:	$\omega^{3},\ \omega^{4},\ \omega^{5}$:	**6,**	**1,**	**7**
2:	$\omega^{6},\ \omega^{7},\ \omega^{8}$:	**11,**	**7,**	**77**

Define a polynomial W: $\Omega \rightarrow \Omega$ that implements a rotation:

$$W(\omega^{-2}, \omega^{1}, \omega^{3}) = (\omega^{1}, \omega^{3}, \omega^{-2})\ , \quad W(\omega^{-1}, \omega^{0}) = (\omega^{0}, \omega^{-1})\ , \ \ldots$$

<u>**Lemma**</u>: $\forall\, y \in \Omega$: $T(y) = T(W(y))$ $\Rightarrow$ wire constraints are satisfied

Proving (3): T respects the wires of C

Copy constraints:

$T(\omega^{-2}) = T(\omega^{1}) = T(\omega^{3})$

$T(\omega^{-1}) = T(\omega^{0})$

$T(\omega^{2}) = T(\omega^{6})$

example: $x_1=5,\ x_2=6\ ,\ w_1=1$

	$\omega^{-1},\ \omega^{-2},\ \omega^{-3}$:	**5,**	**6,**	**1**
0:	$\omega^{0},\ \omega^{1},\ \omega^{2}$:	**5,**	**6,**	**11**
1:	$\omega^{3},\ \omega^{4},\ \omega^{5}$:	**6,**	**1,**	**7**
				77

Proved using a prescribed permutation check

Define a polynom … $\rightarrow \Omega$ that implements a rotation:

$W(\omega^{-2}, \omega^{1}, \omega^{3})$ … $\omega^{3}, \omega^{-2})$, $W(\omega^{-1}, \omega^{0}) = (\omega^{0}, \omega^{-1})$, …

<u>**Lemma**</u>: $\forall\ y \in \Omega$: $T(y) = T(W(y)) \ \Rightarrow$ wire constraints are satisfied

The complete Plonk Poly-IOP (and SNARK)

Setup(C) → $pp := (S, W)$ and $vp := (\boxed{S}$ and $\boxed{W})$ (untrusted)

<u>Prover P($pp, \boldsymbol{x}, \mathbf{w}$)</u>

build $T(X) \in \mathbb{F}_p^{(\leq d)}[X]$ $\xrightarrow{\boxed{T}}$

<u>Verifier V($vp, \boldsymbol{x}$)</u>

build $v(X) \in \mathbb{F}_p^{(\leq |I_x|)}[X]$

Prover proves:

gates: (1) $S(y) \cdot [T(y) + T(\omega y)] + (1 - S(y)) \cdot T(y) \cdot T(\omega y) - T(\omega^2 y) = 0$; $\forall\, y \in \Omega_{\text{gates}}$

inputs: (2) $T(y) - v(y) = 0$ $\forall\, y \in \Omega_{\text{inp}}$

wires: (3) $T(y) - T(W(y)) = 0$ (using prescribed perm. check) $\forall\, y \in \Omega$

output: (4) $T(\omega^{3|C|-1}) = 0$ (output of last gate = 0)

The complete Plonk Poly-IOP (and SNARK)

Setup(C) → $pp := (S, W)$ and $vp := (\boxed{S}$ and $\boxed{W})$ (untrusted)

<u>Prover P($pp, \boldsymbol{x}, \mathbf{w}$)</u>

build $T(X) \in \mathbb{F}_p^{(\le d)}[X]$

$\xrightarrow{\boxed{T}}$

<u>Verifier V($vp, \boldsymbol{x}$)</u>

build $v(X) \in \mathbb{F}_p^{(\le |I_x|)}[X]$

<u>Thm</u>: The Plonk Poly-IOP is complete and knowledge sound, assuming $7|C|/p$ is negligible

(eprint/2019/953)

Many extensions …

Plonk proof: a short proof (O(1) commitments), fast verifier

The SNARK can be made into a zk-SNARK

Main challenge: reduce prover time

- **Hyperplonk**: replace Ω with $\{0,1\}^t$ (where $t = \log_2|\Omega|$)
  - The polynomial T is now a multilinear polynomial in t variables
  - ZeroTest is replaced by a multilinear SumCheck (linear time)

END OF LECTURE

Next lecture: scaling the blockchain