Zero Knowledge Proofs

The Plonk SNARK

Instructors: Dan Boneh, Shafi Goldwasser, Dawn Song, Justin Thaler, Yupeng Zhang
Let’s build an efficient SNARK

- A polynomial commitment scheme
- A polynomial interactive oracle proof (IOP)
- SNARK for general circuits
First, a review of polynomial commitments

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

- **eval**: for public $u, v \in \mathbb{F}_p$, prover can convince the verifier that committed poly satisfies
  
  $f(u) = v$ and $\deg(f) \leq d$.  

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

  verifier has $(d, \text{com}_f, u, v)$
The KZG poly-commit scheme  (Kate-Zaverucha-Goldberg’2010)

Group $\mathbb{G} := \{0, G, 2 \cdot G, 3 \cdot G, \ldots, (p-1) \cdot G\}$ of order $p$.

**setup**$(1^\lambda) \rightarrow gp:$
- Sample random $\tau \in \mathbb{F}_p$
- $gp = \left( H_0 = G, \ H_1 = \tau \cdot G, \ H_2 = \tau^2 \cdot G, \ldots, \ H_d = \tau^d \cdot G \right) \in \mathbb{G}^{d+1}$
- delete $\tau$ !! (trusted setup)

**commit**$(gp, f)$ $\rightarrow$ $com_f$ where $com_f := f(\tau) \cdot G \in \mathbb{G}$

- $f(X) = f_0 + f_1 X + \cdots + f_d X^d \Rightarrow com_f = f_0 \cdot H_0 + \cdots + f_d \cdot H_d$

$$= f_0 \cdot G + f_1 \tau \cdot G + f_2 \tau^2 \cdot G + \cdots = f(\tau) \cdot G$$
The KZG poly-commit scheme (Kate-Zaverucha-Goldberg’2010)

Group $\mathbb{G} := \{ 0, G, 2 \cdot G, 3 \cdot G, \ldots, (p-1) \cdot G \}$ of order $p$.

**setup**$(1^\lambda) \rightarrow gp$:
- Sample random $\tau \in \mathbb{F}_p$
- $gp = \left( H_0 = G, \ H_1 = \tau \cdot G, \ H_2 = \tau^2 \cdot G, \ldots, \ H_d = \tau^d \cdot G \right) \in \mathbb{G}^{d+1}$
- delete $\tau$ !! (trusted setup)

**commit**$(gp, f) \rightarrow com_f$ where $com_f := f(\tau) \cdot G \in \mathbb{G}$

- $f(X) = f_0 + f_1X + \cdots + f_dX^d \Rightarrow com_f = f_0 \cdot H_0 + \cdots + f_d \cdot H_d$

  $= f_0 \cdot G + f_1\tau \cdot G + f_2\tau^2 \cdot G + \cdots = f(\tau) \cdot G$

A binding commitment, but not hiding.
eval: \[ \begin{align*} 
\text{Prover}(gp, f, u, v) & \quad \text{Goal: prove } f(u) = v \\
\text{Verifier}(gp, \text{com}_f, u, v) & 
\end{align*} \]

\[ f(u) = v \iff u \text{ is a root of } \hat{f} := f - v \iff (X-u) \text{ divides } \hat{f} \]

\[ \iff \exists q \in \mathbb{F}_p[X] \text{ s.t. } q(X) \cdot (X-u) = f(X) - v \]

compute \( q(X) \) and \( \text{com}_q = q(\tau) \cdot G \)

\( \pi := \text{com}_q \in \mathbb{G} \) (proof size indep. of deg. d) accept if

\( (\tau - u) \cdot \text{com}_q = \text{com}_f - v \cdot G \)
The KZG poly-commit scheme  
(Kate-Zaverucha-Goldberg’2010)

\[ \text{commit}(gp, f) \rightarrow \text{com}_f \quad \text{where} \quad \text{com}_f = f(\tau) \cdot G \in \mathbb{G} \]

**eval:**

- **Prover** \((gp, f, u, v)\)
  - Goal: prove \(f(u) = v\)
  - compute \(q(X)\) and \(\text{com}_q = q(\tau) \cdot G\)
  - (proof size indep. of deg. \(d\))

- **Verifier** \((gp, \text{com}_f, u, v)\)
  - divides \(\hat{f}\)
  - \(\exists q(X) \in \mathbb{F}_p[X] \text{ s.t. } q(X) \cdot (X-u) = f(X) - v\)
  - accept if \((\tau - u) \cdot \text{com}_q = \text{com}_f - v \cdot G\) (proof size indep. of deg. \(d\))
The KZG poly-commit scheme (Kate-Zaverucha-Goldberg’2010)

How to prove that this is a secure PCS? Not today ...

**eval:** $\text{Prover}(gp,f,u,v)$

Goal: prove $f(u) = v$

Verifier does not know $\tau \Rightarrow$ uses a “pairing” (and only needs $H_0, H_1$ from $gp$)

compute $q(X)$ and $\text{com}_q = q(\tau) \cdot G$

(proof size indep. of deg. $d$)

$\pi := \text{com}_q \in \mathbb{G}$

accept if $(\tau - u) \cdot \text{com}_q = \text{com}_f - v \cdot G$
The KZG poly-commit scheme  
(Kate-Zaverucha-Goldberg’2010)

Generalizations:

- Can also use KZG to commit to \textit{k-variate polynomials} \cite{PST13}

- \textbf{Batch proofs}:
  
  - suppose verifier has commitments \( \text{com}_{f_1}, \ldots, \text{com}_{f_n} \)
  
  - prover wants to prove \( f_i(u_{i,j}) = v_{i,j} \) for \( i \in [n], j \in [m] \)

  \[ \Rightarrow \text{ batch proof } \pi \text{ is only one group element !} \]
Properties of KZG: linear time commitment

Two ways to represent a polynomial \( f(X) \) in \( \mathbb{F}^{(\leq d)}_p[X] \):

- **Coefficient representation:** \( f(X) = f_0 + f_1X + \cdots + f_dX^d \)
  
  \( \Rightarrow \) computing \( \text{com}_f = f_0 \cdot H_0 + \cdots + f_d \cdot H_d \) takes linear time in \( d \)

- **Point-value representation:** \( (a_0, f(a_0)), \ldots, (a_d, f(a_d)) \)
  
  computing \( \text{com}_f \) naively: construct coefficients \( (f_0, f_1, \ldots, f_d) \)
  
  \( \Rightarrow \) time \( O(d \log d) \) using Num. Th. Transform (NTT)
Properties of KZG: linear time commitment

**Point-value representation:** a better way to compute $\text{com}_f$

Lagrange interpolation: $f(\tau) = \sum_{i=0}^{d} \lambda_i(\tau) \cdot f(a_i)$

where

$$ \lambda_i(\tau) = \frac{\prod_{j=0, j \neq i}^{d} (\tau - a_j)}{\prod_{j=0, j \neq i}^{d} (a_i - a_j)} \in \mathbb{F}_p $$

- **Idea:** transform $gp$ into Lagrange form (a linear map)

$$ \hat{gp} = \left( \hat{H}_0 = \lambda_0(\tau) \cdot G, \ \hat{H}_1 = \lambda_1(\tau) \cdot G, \ \ldots, \ \hat{H}_d = \lambda_d(\tau) \cdot G \right) \in \mathbb{G}^{d+1} $$

- **Now,**

$$ \text{com}_f = f(\tau) \cdot G = f(a_0) \cdot \hat{H}_0 + \cdots + f(a_d) \cdot \hat{H}_d $$

$\Rightarrow$ linear time in $d$. (better than $O(d \log d)$)
Prover has some $f(X)$ in $\mathbb{F}_p^{(\leq d)}[X]$. Let $\Omega \subseteq \mathbb{F}_p$ and $|\Omega| = d$

Suppose prover needs evaluation proofs $\pi_a \in G$ for all $a \in \Omega$

- Naively, takes time $O(d^2)$: $d$ proofs each takes time $O(d)$
- **Feist-Khovratovich** (FK) algorithm (2020):
  - if $\Omega$ is a multiplicative subgroup: time $O(d \log d)$
  - otherwise: time $O(d \log^2 d)$
Difficulties with KZG: trusted setup for \( gp \), and \( gp \) size is linear in \( d \).

**Dory:**
- **transparent setup:** no secret randomness in setup
- \( com_f \) is a single group element (independent of degree \( d \))
- eval proof size for \( f \in \mathbb{F}_p^{(\leq d)}[X] \) is \( O(\log d) \) group elements
- eval verify time is \( O(\log d) \)
- Prover time: \( O(d) \)
**PCS have many applications**

**Example: vector commitment** (a drop-in replacement for Merkle trees)

<table>
<thead>
<tr>
<th>Bob: vector $(u_1, \ldots, u_k) \in \mathbb{F}_p^{(\leq d)}$</th>
<th>Alice</th>
</tr>
</thead>
<tbody>
<tr>
<td>interpolate poly $f$ s.t.: $f(i) = u_i$ for $i = 1, \ldots, k$</td>
<td>$\text{com}_f := \text{commit}(gp, f)$</td>
</tr>
<tr>
<td>$\pi := \text{eval proof that } f(2) = a, f(4) = b$ (KZG: $\pi$ is a single group element) shorter than a Merkle proof!</td>
<td>prove $u_2 = a$, $u_4 = b$</td>
</tr>
<tr>
<td></td>
<td>$\pi \in \mathbb{G}$ accept or reject</td>
</tr>
</tbody>
</table>
Proving properties of committed polynomials
Proving properties of committed polynomials

**Prover** $P(f, g)$

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

**Verifier** $V(f, g)$

Proof systems presented as an IOP:

- $r \leftarrow \mathbb{F}_p$
- $q$  
- Query $f(X), g(X), q(X)$ at some points in $\mathbb{F}_p$  
  - [V sends $x$ to P who responds with $f(x)$ and eval proof $\pi$]

accept or reject
Recall: polynomial equality testing

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.

\[
f(r) - g(r) = 0 \ \Rightarrow \ f - g = 0 \ \text{w.h.p}
\]

\( \Rightarrow \) a simple equality test for two committed polynomials.
Review: the proof system as an IOP

Prover

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

Verifier

\[ r \leftarrow \mathbb{F}_p \]

learn \( f(r), g(r) \)

accept if:

\[ f(r) = g(r) \]
Review: the compiled proof system

Prover

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

\( y \leftarrow f(r) \)

\( y' \leftarrow g(r) \)

Verifier

\( r \leftarrow \mathbb{F}_p \)

Accept if:

(i) \( y = y' \), and

(ii) \( \pi_f, \pi_g \) are valid

Proof that

\( y = f(r) \)

Proof that

\( y' = g(r) \)
Review: the compiled proof system

Prover

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

\( y \leftarrow f(r) \)

\( y' \leftarrow g(r) \)

A public coin protocol

Make non-interactive using Fiat-Shamir

Verifier

\( r \leftarrow \mathbb{F}_p \)

accept if:

(i) \( y = y' \), and

(ii) \( \pi_f, \pi_g \) are valid

proof that \( y = f(r) \)

proof that \( y' = g(r) \)
Polynomial equality testing with KZG

For KZG: \[ f = g \iff \text{com}_f = \text{com}_g \]

\[ \implies \text{verifier can tell if } f = g \text{ on its own} \]

But prover is needed to test equality of computed polynomials

- Example: verifier has \( f, g_1, g_2, g_3 \) where all four are in \( \mathbb{F}_p^{(\leq d)}[X] \)

  to test if \( f = g_1g_2g_3 \): V queries all four poly. at \( r \leftarrow \mathbb{F}_p \) and tests equality

- Complete and sound assuming \( 3d/p \) is negligible \( \quad (\deg(g_1g_2g_3) \leq 3d) \)
Important proof gadgets for univariates

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

Let $f \in \mathbb{F}_p^{(\leq d)}[X]$ \hspace{1cm} (d \geq k) \hspace{1cm} \text{Verifier has} \hspace{1cm} [f]

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

Task 1 (ZeroTest): prove that $f$ is identically zero on $\Omega$
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 $\Omega$ be some subset of $\mathbb{F}_p$ of size $k$.

**Def:** the *vanishing polynomial* of $\Omega$ 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 $\Omega = \{1, \omega, \omega^2, ..., \omega^{k-1}\} \subseteq \mathbb{F}_p$ then $Z_\Omega(X) = X^k - 1$

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

$\Omega = \{ 1, \omega, \omega^2, \ldots, \omega^{k-1} \}$

**Prover P($f$)**

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

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

**Verifier V($f$)**

$r \leftarrow \mathbb{F}_p$  

$r \rightharpoonup f(r)$  

verify $q(r) \cdot Z_\Omega(r)$

**Lemma:** $f$ is zero on $\Omega$ 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 $\Omega$  \hspace{1cm} ($\Omega = \{1, \omega, \omega^2, \ldots, \omega^{k-1}\}$)

**Prover P($f$)**

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

**Verifier V($f$)**

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

$\begin{align*}
q(X) &\leftrightarrow f(X)/Z_\Omega(X) \\
q \in &\mathbb{F}_p^{(\leq d)} [X] \\
\end{align*}$

Verifier time: $O(\log k)$ and two poly queries (but can be done in one)

Prover time: dominated by the time to compute $q(X)$ and then commit to $q(X)$

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

$q \leftarrow \mathbb{F}_p$

Learn $q(r), f(r)$

Accept if $f(r) \equiv q(r) \cdot Z_\Omega(r)$
(3) Product check on $\Omega$ : $\prod_{a \in \Omega} f(a) = 1$

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

\[
\begin{align*}
    t(1) &= f(1), \\
    t(\omega^s) &= \prod_{i=0}^{s} f(\omega^i) \quad \text{for } s = 1, \ldots, k - 1
\end{align*}
\]

Then

\[
\begin{align*}
    t(\omega) &= f(1) \cdot f(\omega), \\
    t(\omega^2) &= f(1) \cdot f(\omega) \cdot f(\omega^2), \quad \ldots \\
    t(\omega^{k-1}) &= \prod_{a \in \Omega} f(a) = 1
\end{align*}
\]

and

\[
\begin{align*}
    t(\omega \cdot x) &= t(x) \cdot f(\omega \cdot x) \quad \text{for all } x \in \Omega \quad (\text{including at } x = \omega^{k-1})
\end{align*}
\]
(3) Product check on $\Omega$ : $\prod_{a \in \Omega} f(a) = 1$

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

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

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

(ii) $t(\omega \cdot x) - t(x) \cdot f(\omega \cdot x) = 0$ for all $x \in \Omega$

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

**Prover** $P(f)$

- Construct $t(X) \in \mathbb{F}_p^{(\leq k)}$ and $t_1(X) = t(\omega \cdot X) - t(X) \cdot f(\omega \cdot X)$
- Set $q(X) = t_1(X)/(X^k - 1) \in \mathbb{F}_p^{(\leq d)}$

**Verifier** $V([f])$

- $t_1(X)$ should be zero on $\Omega$
- $r \xleftarrow{\$} \mathbb{F}_p$
- Learn $t(\omega^{k-1})$, $t(r)$, $t(\omega r)$, $q(r)$, $f(\omega r)$

Query $t(X)$ at $\omega^{k-1}$, $r$, $\omega r$

Query $q(X)$ at $r$, and $f(X)$ at $\omega r$

Proves that $t_1(\Omega) = 0$:

- $t_1(\Omega) = 0$
(3) Product check on $\Omega$  (unoptimized)

**Prover P($f$)**

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

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

**Verifier V([f])**

$r \leftarrow \mathbb{F}_p$

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

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

A public coin protocol

Proof size: two commits, five evals. Verifier time: $O(\log k)$. Prover time: $O(k \log k)$. 
Same works for rational functions: \( \prod_{a \in \Omega} (f/g)(a) = 1 \)

**Prover** \( P(f, g) \)

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

\[
\begin{align*}
    t(1) &= f(1)/g(1), \\
    t(\omega^s) &= \prod_{i=0}^{s} f(\omega^i)/g(\omega^i) \quad \text{for} \quad s = 1, \ldots, k - 1
\end{align*}
\]

**Verifier** \( V(f, g) \)

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

**Goal:** prover wants to prove that 
\[
\begin{pmatrix}
f(1), f(\omega), f(\omega^2), \ldots, f(\omega^{k-1})
g(1), g(\omega), g(\omega^2), \ldots, g(\omega^{k-1})
\end{pmatrix}
\in \mathbb{F}_p^k
\]

is a permutation of 
\[
\begin{pmatrix}
f(1), f(\omega), f(\omega^2), \ldots, f(\omega^{k-1})
g(1), g(\omega), g(\omega^2), \ldots, g(\omega^{k-1})
\end{pmatrix}
\in \mathbb{F}_p^k
\]

$\Rightarrow$ Proves that $g(\Omega)$ is the same as $f(\Omega)$, just permuted
(4) Another useful gadget: permutation check

Prover P(\(f, g\))

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

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

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

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

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

Verifier V(\(f, g\))

A public coin protocol

\(r \leftarrow \mathbb{F}_p\)

accept or reject

[Lipton’s trick, 1989]

[two commits, six evals]
(5) final gadget: prescribed permutation check

\[ W : \Omega \rightarrow \Omega \text{ is a permutation of } \Omega \text{ if } \forall i \in [k]: W(\omega^i) = \omega^j \text{ is a bijection} \]

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

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

**Goal:** 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 zero-test to prove \( f(y) - g(W(y)) = 0 \) on \( \Omega \)

The problem: the polynomial \( f(y) - g(W(y)) \) has degree \( k^2 \)

\( \Rightarrow \) prover would need to manipulate polynomials of degree \( k^2 \)

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

Let’s reduce this to a prod-check on a polynomial of degree \( 2k \) (not \( k^2 \))
Prescribed permutation check

Observation:

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

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: \((\omega^2, f(\omega^0)), (\omega^0, f(\omega^1)), (\omega^1, f(\omega^2))\)
Prescribed permutation check

Prover $P(f, g, W)$

Let

\[
\begin{align*}
\hat{f}(X, Y) &= \prod_{a \in \Omega} (X - Y \cdot W(a) - f(a)) \\
\hat{g}(X, Y) &= \prod_{a \in \Omega} (X - Y \cdot a - g(a))
\end{align*}
\]

(bivariate polynomials of total degree $k$)

**Lemma:** $\hat{f}(X, Y) = \hat{g}(X, Y) \iff (W(a), f(a))_{a \in \Omega}$ is a perm. of $(a, g(a))_{a \in \Omega}$

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

**Prover** $P(f, g, W)$

- $r, s$

**Verifier** $V(f, g, W)$

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

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

ProdCheck: \[
\prod_{a \in \Omega} \left( \frac{r - s \cdot W(a) - f(a)}{r - s \cdot a - g(a)} \right) = 1
\]

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

Complete and sound, assuming $2d/p$ is negligible.
Summary of proof gadgets

- Polynomial equality testing
- Zero test on $\Omega$
- Product check, sum check
- Permutation check
- Prescribed permutation check
The PLONK IOP for general circuits

eprint/2019/953
PLONK: widely used in practice

- The Plonk IOP
  - Polynomial commitment scheme
    - KZG’10 (pairings)
    - Bulletproofs (no pairings)
    - FRI (hashing)
  - SNARK system
    - Aztec, JellyFish
    - Halo2 (slow verifier, no trusted setup)
    - Plonky2 (no trusted setup)
PLONK: a poly-IOP for a general circuit $C(x, w)$

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

The computation trace (arithmetization):

<table>
<thead>
<tr>
<th>inputs:</th>
<th>5, 6, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0:</td>
<td>5, 6, 11</td>
</tr>
<tr>
<td>Gate 1:</td>
<td>6, 1, 7</td>
</tr>
<tr>
<td>Gate 2:</td>
<td>11, 7, 77</td>
</tr>
</tbody>
</table>

Example input:

- $x_1$: 5
- $x_2$: 6
- $w_1$: 1
Encoding the trace as a polynomial

\[ |C| := \text{total \# of gates in } C, \quad |I| := |I_x| + |I_w| = \# \text{ inputs to } C \]

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

The plan:

prover interpolates a polynomial \( T \in \mathbb{F}_p^{(\leq d)}[X] \) that encodes the entire trace.

Let's see how ...

<table>
<thead>
<tr>
<th>inputs:</th>
<th>5, 6, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0:</td>
<td>5, 6, 11</td>
</tr>
<tr>
<td>Gate 1:</td>
<td>6, 1, 7</td>
</tr>
<tr>
<td>Gate 2:</td>
<td>11, 7, 77</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>inputs:</th>
<th>5, 6, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0:</td>
<td>5, 6, 11</td>
</tr>
<tr>
<td>Gate 1:</td>
<td>6, 1, 7</td>
</tr>
<tr>
<td>Gate 2:</td>
<td>11, 7, 77</td>
</tr>
</tbody>
</table>
Encoding the trace as a polynomial

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

<table>
<thead>
<tr>
<th>inputs:</th>
<th>$T(\omega^{-1}) = 5$,</th>
<th>$T(\omega^{-2}) = 6$,</th>
<th>$T(\omega^{-3}) = 1$,</th>
</tr>
</thead>
<tbody>
<tr>
<td>gate 0:</td>
<td>$T(\omega^0) = 5$,</td>
<td>$T(\omega^1) = 6$,</td>
<td>$T(\omega^2) = 11$,</td>
</tr>
<tr>
<td>gate 1:</td>
<td>$T(\omega^3) = 6$,</td>
<td>$T(\omega^4) = 1$,</td>
<td>$T(\omega^5) = 7$,</td>
</tr>
<tr>
<td>gate 2:</td>
<td>$T(\omega^6) = 11$,</td>
<td>$T(\omega^7) = 7$,</td>
<td>$T(\omega^8) = 77$</td>
</tr>
</tbody>
</table>

$\text{degree}(T) = 11$

Prover can use FFT to compute the coefficients of $T$ in time $O(d \log d)$
Step 2: proving validity of T

Prover $P(S_p, x, w)$  
Verifier $V(S_v, x)$

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

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)

<table>
<thead>
<tr>
<th>inputs:</th>
<th>5, 6, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0:</td>
<td>5, 6, 11</td>
</tr>
<tr>
<td>Gate 1:</td>
<td>6, 1, 7</td>
</tr>
<tr>
<td>Gate 2:</td>
<td>11, 7, 77</td>
</tr>
</tbody>
</table>
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:

$$v(\omega^{-j}) = \text{input #}j$$

In our example: $v(\omega^{-1}) = 5, \ v(\omega^{-2}) = 6$. ($v$ 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 prover and verifier interpolate a polynomial \( v(X) \in \mathbb{F}_p^{(\leq |I_x|)}[X] \) that encodes the \( x \)-inputs to the circuit:

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

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 \( \Omega_{\text{inp}} \) to prove that

\[
T(y) - v(y) = 0 \quad \forall \ y \in \Omega_{\text{inp}}
\]
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 $\forall \ l = 0, \ldots, |C| - 1$:

- $S(\omega^{3l}) = 1$ if gate $\#l$ is an addition gate
- $S(\omega^{3l}) = 0$ if gate $\#l$ is a multiplication gate

<table>
<thead>
<tr>
<th>inputs:</th>
<th>5, 6, 1</th>
<th>$S(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0 ($\omega^0$): 5, 6, 11</td>
<td>1</td>
<td>(+)</td>
</tr>
<tr>
<td>Gate 1 ($\omega^3$): 6, 1, 7</td>
<td>1</td>
<td>(+)</td>
</tr>
<tr>
<td>Gate 2 ($\omega^6$): 11, 7, 77</td>
<td>0</td>
<td>(x)</td>
</tr>
</tbody>
</table>
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 $\forall y \in \Omega_{\text{gates}} := \{1, \omega^3, \omega^6, \omega^9, \ldots, \omega^{3(|C|-1)}\}$:

$$S(y) \cdot [T(y) + T(\omega y)] + (1 - S(y)) \cdot T(y) \cdot T(\omega y) = T(\omega^2 y)$$
Proving (2): every gate is evaluated correctly

\[
\begin{align*}
\text{Setup}(C) & \rightarrow pp := S \text{ and } vp := (\overline{S}) \\
\text{Prover } P(pp, x, w) & \text{ build } T(X) \in \mathbb{F}_p^{(\leq d)}[X] \\
\text{Verifier } V(vp, x) &
\end{align*}
\]

Prover uses ZeroTest to prove that for all \( \forall y \in \Omega \) gates:

\[
S(y) \cdot [T(y) + T(\omega y)] + (1 - S(y)) \cdot T(y) \cdot T(\omega y) - T(\omega^2 y) = 0
\]
Proving (3): the wiring is correct

**Step 4:** encode the wires of $C$:

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

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

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

**Example:** $x_1=5$, $x_2=6$, $w_1=1$

$\omega^{-1}, \omega^{-2}, \omega^{-3}: 5, 6, 1$

$0: \omega^0, \omega^1, \omega^2: 5, 6, 11$

$1: \omega^3, \omega^4, \omega^5: 6, 1, 7$

$2: \omega^6, \omega^7, \omega^8: 11, 7, 77$

**Lemma:** $\forall y \in \Omega : T(y) = T(W(y)) \Rightarrow$ wire constraints are satisfied
Proving (3): the wiring is correct

**Step 4:** encode the wires of $C$:

$$
T(\omega^{-2}) = T(\omega^1) = T(\omega^3) \\
T(\omega^{-1}) = T(\omega^0)
$$

**Example:** $x_1=5, x_2=6, w_1=1$

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

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

**Lemma:** $\forall y \in \Omega : T(y) = T(W(y)) \implies$ wire constraints are satisfied
The complete Plonk Poly-IOP (and SNARK)

Setup\((\mathcal{C})\) $\rightarrow$ \(pp := (S,W)\) and \(vp := (\begin{bmatrix} S \end{bmatrix} \text{ and } \begin{bmatrix} W \end{bmatrix})\) (untrusted)

Prover \(P(pp,x,w)\)

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

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

Verifier \(V(vp,x)\)

build \(v(X) \in \mathbb{F}_p^{(\leq |I_x|)}[X]\)
The complete Plonk Poly-IOP (and SNARK)

\[ \text{Setup}(C) \rightarrow \quad pp := (S,W) \quad \text{and} \quad vp := (S \text{ and } W) \]

Prover \( P(pp, x, w) \)

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

Verifier \( V(vp, x) \)

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

**Thm:** 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 easily be made into a zk-SNARK

Main challenge: reduce prover time

- **Hyperplonk**: replace $\Omega$ 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)
A generalization: plonkish arithmetization

Plonk for circuits with gates other than $+$ and $\times$ on rows:

Plonkish computation trace: (also used in AIR)

An example custom gate:

$$\forall y \in \Omega \text{ gates: } v(y\omega) + w(y) \cdot t(y) - t(y\omega) = 0$$

All such gate checks are included in the gate check

Plookup: ensure some values are in a pre-defined list
END OF LECTURE

Next lecture:
More polynomial commitments