# Zero Knowledge Proofs

## The Plonk SNARK

Instructors: **Dan Boneh**, Shafi Goldwasser, Dawn Song, Justin Thaler, Yupeng Zhang





Stanford University
Berkeley UNIVERSITY OF CALIFORNIA
GEORGETOWN UNIVERSITY
TEXAS A&M UNIVERSITY

# 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

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

verifier has $(d, com_f, u, v)$

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

# The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

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

<u>*setup*</u>$(1^\lambda) \rightarrow$ *gp*:

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

<u>*commit*</u>(*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)

a binding commitment, but not hiding

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

- $f(X) = f_0 + f_1 X + \cdots + f_d X^d \quad \Rightarrow \quad \mathbf{com}_f = f_0 \cdot H_0 + \cdots + f_d \cdot H_d$

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

# The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

<u>commit</u>$(gp, f) \rightarrow$ $\mathbf{com}_f$ where $\mathbf{com}_f = f(\tau) \cdot G \in \mathbb{G}$

<u>eval:</u>

<u>Prover(gp, **f**, u, v)</u>

Goal: prove $f(u) = v$

<u>Verifier(gp, $\mathbf{com}_f$, u, v)</u>

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

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

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

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

(proof size indep. of deg. d)

accept if
$(\tau - u)\cdot\mathbf{com}_q = \mathbf{com}_f - v\cdot G$

# The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

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

<u>*eval:*</u>

<u>*Prover(gp, **f**, u, v)*</u>

Goal: prove $f(u) = v$

<u>*Verifier(gp, **com**$_f$, u, v)*</u>

$f(u) = v$ ... divides $\hat{f}$

$\Leftrightarrow$ exists $q$ ... $[X]$ s.t. $q(X) \cdot (X\text{-}u) = f(X) - v$

$$(\tau - u)\, q(\tau) \cdot G \stackrel{?}{=} (f(\tau) - v) \cdot G$$

compute $q(X)$
and $\boldsymbol{com_q} = q(\tau) \cdot G$

$\pi := \boldsymbol{com_q} \in \mathbb{G}$

(proof size indep. of deg. d)

accept if
$(\tau - u) \cdot \boldsymbol{com_q} = \boldsymbol{com_f} - v \cdot G$

# The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

con … $\in \mathbb{G}$

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

eval: *Prover(gp, **f**, u, v)* Goal: prove $f(u) = v$ *Verifier(gp, **com**$_f$, u, v)*

An expensive computation for large $d$

of

$\Leftrightarrow$

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

compute $q(X)$
and $\boldsymbol{com_q} = q(\tau) \cdot G$

$\pi := \boldsymbol{com_q} \in \mathbb{G}$

(proof size indep. of deg. d)

accep…
$(\tau - u) \cdot \boldsymbol{com_q} = \boldsymbol{com_f} - v \cdot G$

# The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

**Generalizations**:

- Can also use KZG to commit to ***k*-variate polynomials** [PST'13]
- **Batch proofs**:
  - suppose verifier has commitments ***com**$_{f1}$* , … ***com**$_{fn}$*
  - prover wants to prove $f_i(u_{i,j}) = v_{i,j}$ for $i \in [n]$, $j \in [m]$

$\Rightarrow$ batch proof $\pi$ is only one group element !

# Properties of KZG: linear time commitment

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

- **Coefficient representation**: $f(X) = f_0 + f_1 X + \cdots + f_d X^d$

  $\Rightarrow$ computing $\boldsymbol{com_f} = f_0 \cdot H_0 + \cdots + f_d \cdot H_d$ takes linear time in $d$

- **Point-value representation**: $(a_0, f(a_0)), \ldots, (a_d, f(a_d))$

  computing $\boldsymbol{com_f}$ 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 $\boldsymbol{com_f}$

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

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

- Now, $\boxed{\boldsymbol{com_f} = f(\tau) \cdot \mathrm{G} = f(a_0) \cdot \widehat{H}_0 + \cdots + f(a_d) \cdot \widehat{H}_d}$

$\Rightarrow$ linear time in $d$. (better than $O(d \log d)$ )

# KZG fast multi-point proof generation

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 $O(d \log^2 d)$

# The Dory polynomial commitment (eprint/2020/1274)

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 $\boxed{\text{O}(\log d)}$ group elements
- eval verify time is $\boxed{\text{O}(\log d)}$ Prover time: $\boxed{O(d)}$

# PCS have many applications

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

**Bob**: vector $(u_1, \dots, u_k) \in \mathbb{F}_p^{(\le d)}$

interpolate poly $f$ s.t.:
$f(i) = u_i$ for $i = 1, \dots, k$

$\mathbf{com}_f :=$ commit$(gp, f)$ →

**Alice**

$\pi :=$ eval proof that $f(2) = a, \; f(4) = b$

(KZG: $\pi$ is a single group element)

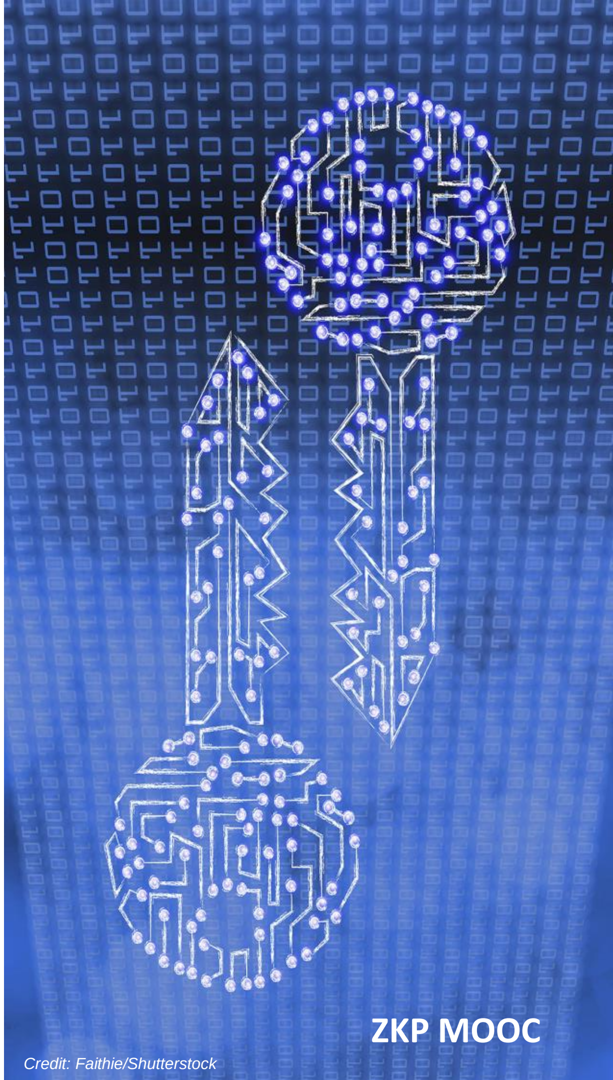
shorter than a Merkle proof!

← prove $u_2 = a, \;\; u_4 = b$

$\pi \in \mathbb{G}$ →

accept or reject

# Proving properties of committed polynomials




*Credit: Faithie/Shutterstock*

# Proving properties of committed polynomials

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

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

Proof systems presented as an IOP:

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

$\boxed{q}$ $\longrightarrow$

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

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

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

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

# Review: the proof system as an IOP

Prover

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

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

Verifier

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

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

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**

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

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

$r$

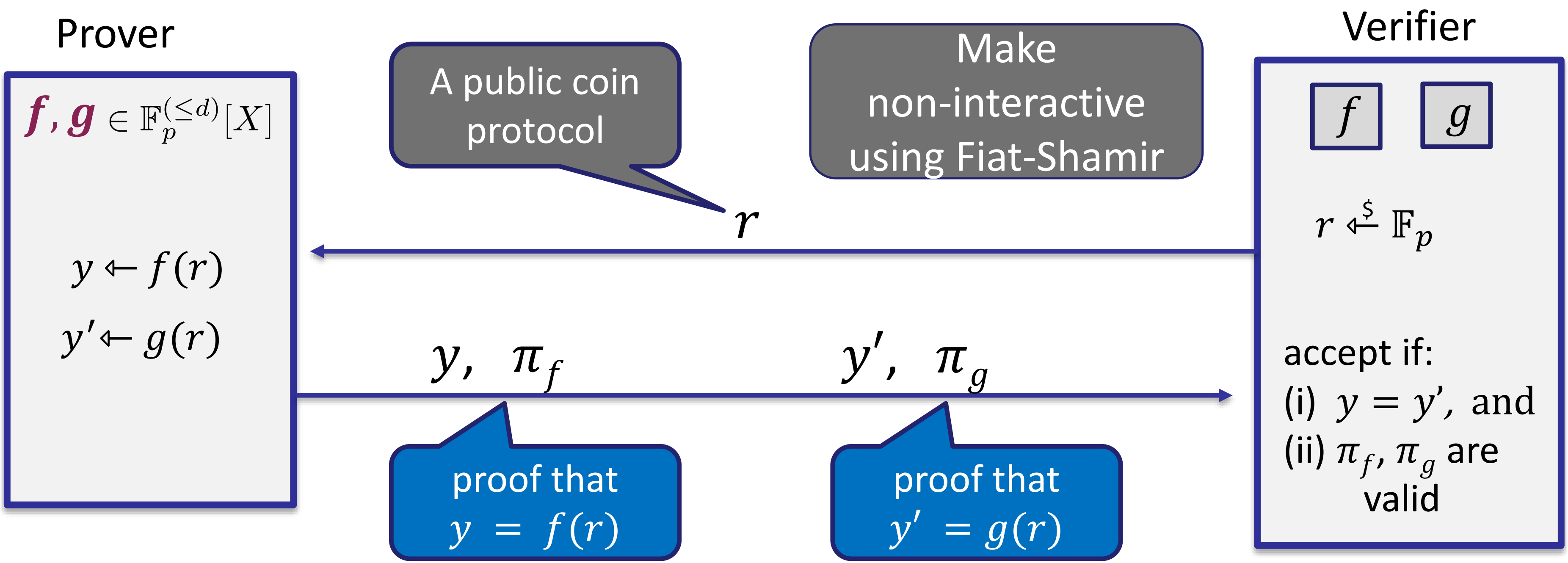
$y, \ \pi_f$ $\qquad$ $y', \ \pi_g$

proof that
$y = f(r)$

proof that
$y' = g(r)$

accept if:
(i) $y = y'$, and
(ii) $\pi_f, \pi_g$ are valid

# Review: the compiled proof system



# Polynomial equality testing with KZG

For KZG: $f = g \iff \mathrm{com}_f = \mathrm{com}_g$

$\Rightarrow$ verifier can tell if $f = g$ on its own

But prover is needed to test equality of computed polynomials

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

  to test if $f = g_1 g_2 g_3$: V queries all four poly. at $r \xleftarrow{\$} \mathbb{F}_p$ and tests equality

- Complete and sound assuming $3d/p$ is negligible ( $\deg(g_1 g_2 g_3) \leq 3d$ )

# Important proof gadgets for univariates

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

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

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 $k$.

<u>Def</u>: 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 $\Omega = \{1, \omega, \omega^2, \ldots, \omega^{k\text{-}1}\} \subseteq \mathbb{F}_p$ 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 $\Omega$ $\quad (\Omega = \{ 1, \omega, \omega^2, ..., \omega^{k-1} \})$

Prover P($f$)

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

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

Verifier V($\boxed{f}$)

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

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

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 $\Omega$ if and only if $f(X)$ is divisible by $Z_\Omega(X)$

**<u>Thm</u>**: this protocol is complete and sound, assuming $d/p$ is negligible.

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

Prover P($f$)

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

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

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

Verifier V($\boxed{f}$)

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

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

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

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

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 $q(X)$

# (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:

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

Then $t(\omega) = f(1) \cdot f(\omega)$, $t(\omega^2) = f(1) \cdot f(\omega) \cdot f(\omega^2)$, …

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

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

# (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:

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

**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($f$)

Verifier V($\boxed{f}$)

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

$t_1(X)$ should be zero on $\Omega$

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

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

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

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

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

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

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

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

# (3) Product check on $\Omega$ (unoptimized)

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

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

A public coin protocol

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

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

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

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

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

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

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

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

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

**Goal**: prover wants to prove that $\big( f(1), f(\omega), f(\omega^2), \ldots, f(\omega^{k-1}) \big) \in \mathbb{F}_p^k$

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

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

A public coin protocol

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

$r$

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

[two commits, six evals]

accept or reject

[Lipton's trick, 1989]

# (5) final gadget: prescribed permutation check

$W : \Omega \to \Omega$ is a **permutation of** $\Omega$ 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 $f, g$ be 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 zero-test to prove $\boxed{f(y) - g(W(y)) = 0 \quad \text{on } \Omega}$

**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 $\big(W(a), f(a)\big)_{a\in\Omega}$ 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: $(\omega^2, f(\omega^0))$, $(\omega^0, f(\omega^1))$, $(\omega^1, f(\omega^2))$

# Prescribed permutation check

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

Let $\begin{cases} \hat{f}(X,Y) = \prod_{a\in\Omega} (X - Y \cdot W(a) - f(a)) \\ \hat{g}(X,Y) = \prod_{a\in\Omega} (X - Y \cdot a - g(a)) \end{cases}$ and

(bivariate polynomials of total degree $k$)

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

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

# The complete protocol

Prover P($f, g, W$)

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

$r, s$

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

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

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

by Schwartz-Zippel

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

Complete and sound, assuming $2d/p$ is negligible.

accept or reject

# Summary of proof gadgets

polynomial equality testing

zero test on $\Omega$

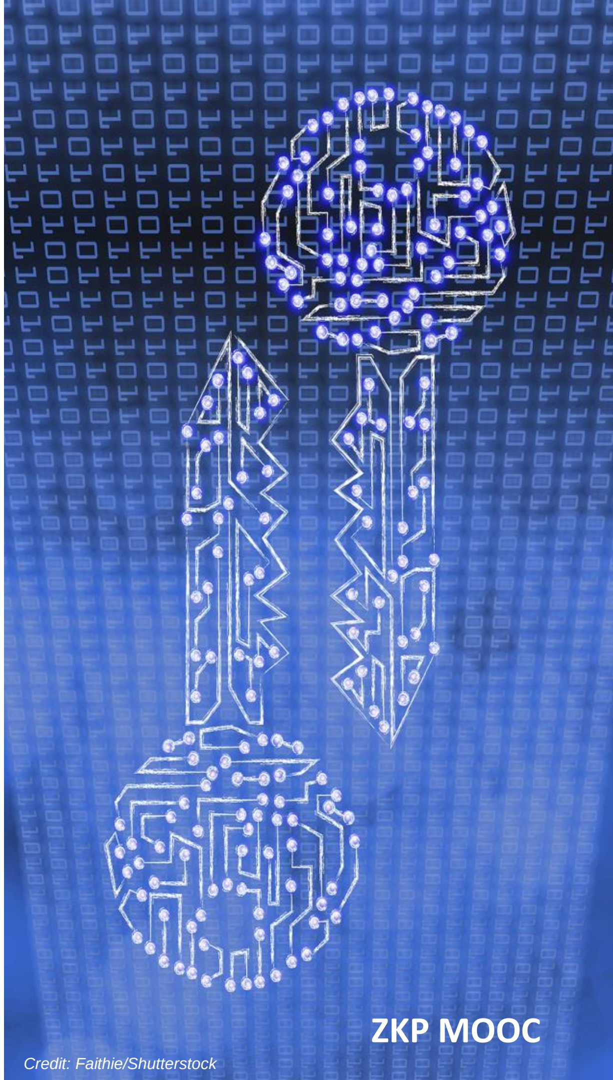
product check, sum check

permutation check

prescribed permutation check

# The PLONK IOP for general circuits

eprint/2019/953

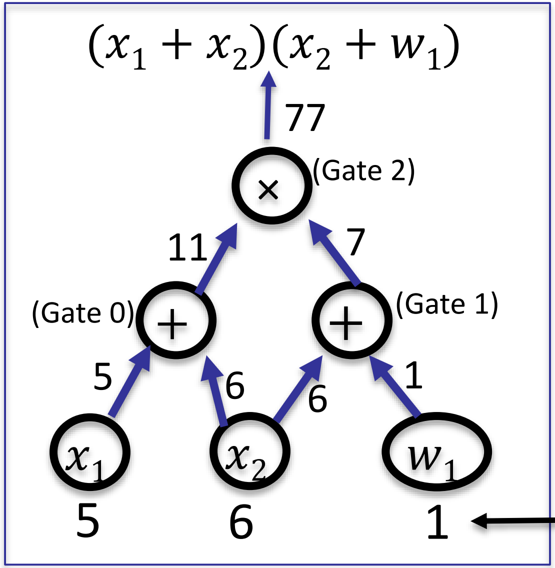


# PLONK: widely used in practice

| | polynomial commitment scheme | SNARK system |
|---|---|---|
| The Plonk IOP | KZG'10 (pairings) | Aztec, JellyFish |
| | Bulletproofs (no pairings) | Halo2 (slow verifier) (no trusted setup) |
| | FRI (hashing) | 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)




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

ZKP MOOC

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

that encodes the entire trace.

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

Let's see how …

# 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

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

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

# Step 2: proving validity of T

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

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

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

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

(wiring constraints)

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

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

# 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: $v(\omega^{-1}) = 5$, $v(\omega^{-2}) = 6$. ($v$ 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:

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

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 $\Omega_{\text{inp}}$ to prove that

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

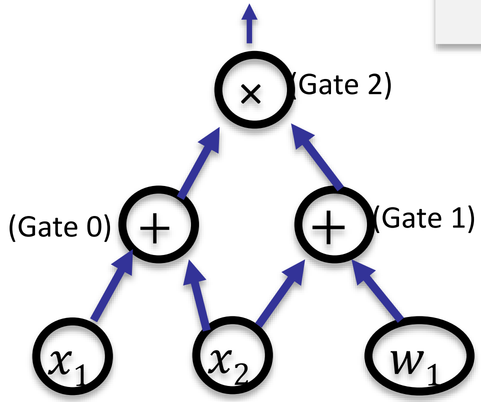
# Proving (2): every gate is evaluated correctly

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

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




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

# Proving (2): every gate is evaluated correctly

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

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

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

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

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

$$S(y) \cdot \big[\underbrace{T(y)}_{\text{left input}} + \underbrace{T(\omega y)}_{\text{right input}}\big] + (1 - S(y)) \cdot \underbrace{T(y)}_{\text{left input}} \cdot \underbrace{T(\omega y)}_{\text{right input}} = \underbrace{T(\omega^2 y)}_{\text{output}}$$

# Proving (2): every gate is evaluated correctly

$\text{Setup}(C) \rightarrow pp := \text{S}$ and $vp := (\boxed{\text{S}})$

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

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

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

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

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

# Proving (3): the wiring is correct

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

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

<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: $\Omega \to \Omega$ 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): 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: $x_1=5$, $x_2=6$, $w_1=1$

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

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

**1, 7**

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

Proved using a prescribed permutation check

Define a polynomia… $\to \Omega$ that implements a rotation:

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

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

# The complete Plonk Poly-IOP (and SNARK)

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

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

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

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

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

Prover proves:

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

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

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

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

# The complete Plonk Poly-IOP (and SNARK)

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

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

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

build T$(X) \in \mathbb{F}_p^{(\leq d)}$[X] $\xrightarrow{\boxed{T}}$ build $v(X) \in \mathbb{F}_p^{(\leq |I_x|)}$ [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 × on rows:

## Plonkish computation trace: (also used in AIR)

An example custom gate:

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

All such gate checks are included in the gate check

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

| | | | | | |
|---|---|---|---|---|---|
| u1 | v1 | w1 | t1 | r1 | s1 |
| u2 | v2 | w2 | t2 | r2 | s2 |
| u3 | v3 | w3 | t3 | r3 | s3 |
| u4 | v4 | w4 | t4 | r4 | s4 |
| u5 | v5 | w5 | t5 | r5 | s5 |
| u6 | v6 | w6 | t6 | r6 | s6 |
| u7 | v7 | w7 | t7 | r7 | s7 |
| u8 | v8 | w8 | t8 | r8 | s8 |

output

# END OF LECTURE

Next lecture:

More polynomial commitments