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

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

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

setup$(1^\lambda) \to gp$:

- 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 τ !! (trusted setup)

commit$(gp, f) \to \mathbf{com}_f$ 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)

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

setup$(1^\lambda) \to gp$:

- 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 τ !! (trusted setup)

a binding commitment, but not hiding

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

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

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

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

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

eval: *Prover(gp, **f**, u, v)* Goal: prove $f(u) = v$ *Verifier(gp, $\boldsymbol{com_f}$, u, v)*

$f(u) = v \iff u$ 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 $\boldsymbol{com_q} = q(\tau)\cdot G$

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

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

eval: *Prover(gp, **f**, u, v)* Goal: prove $f(u) = v$ *Verifier(gp, $\boldsymbol{com_f}$, u, v)*

$f(u) = v$ $\Leftrightarrow$ $f(X) - v$ is divisible by $(X - u)$ $\Leftrightarrow$ exists $q \in \mathbb{F}^{<d}[X]$ s.t. $q(X)\cdot(X-u) = f(X)-v$

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

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)

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

<u>com</u> ∈ 𝔾

<u>*eval*:</u> <u>*Prover(gp, **f**, u, v)*</u> <u>*Verifier(gp, **com**$_f$, u, v)*</u>

Goal: prove $f(u) = v$

An expensive computation for large d

of f

$\Leftrightarrow$

Verifier does not know τ $\Rightarrow$ 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)

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

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 π is only one group element !

Batch opening protocol: an example

Prover P(gp, f, g, x_1, x_2)

Verifier V(gp, $\boxed{f}$, $\boxed{g}$, x_1, x_2, y_1, y_2, y_3)

Goal: convince verifier that $f(x_1) = y_1$, $f(x_2) = y_2$, $g(x_2) = y_3$
by committing to a single polynomial

Homework: show how to do it

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**: $\big(a_0, f(a_0)\big), \ldots, \big(a_d, f(a_d)\big)$

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

- **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, $\boxed{com_f = f(\tau) \cdot G = f(a_0) \cdot \hat{H}_0 + \cdots + f(a_d) \cdot \hat{H}_d}$

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

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 Ω is a multiplicative subgroup: time $O(d \log d)$
  - otherwise: time $O(d \log^2 d)$

The Dory polynomial commitment (eprint/2020/1274)

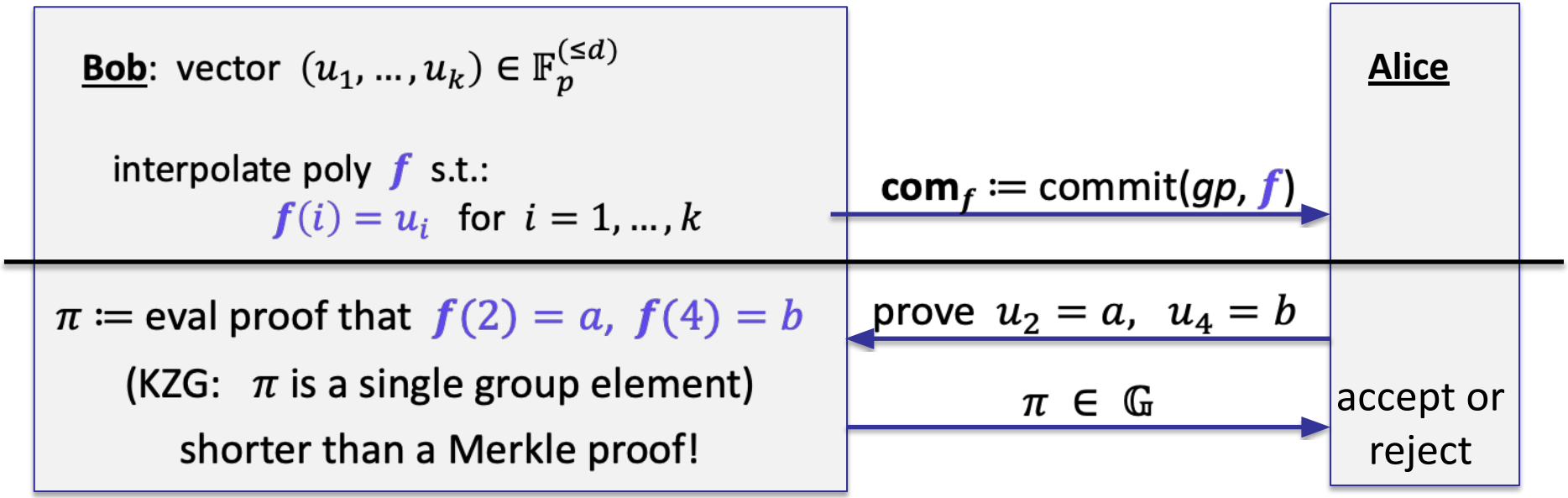
Difficulties with KZG: trusted setup for *gp,* and *gp* size is linear in *d*.

Dory:

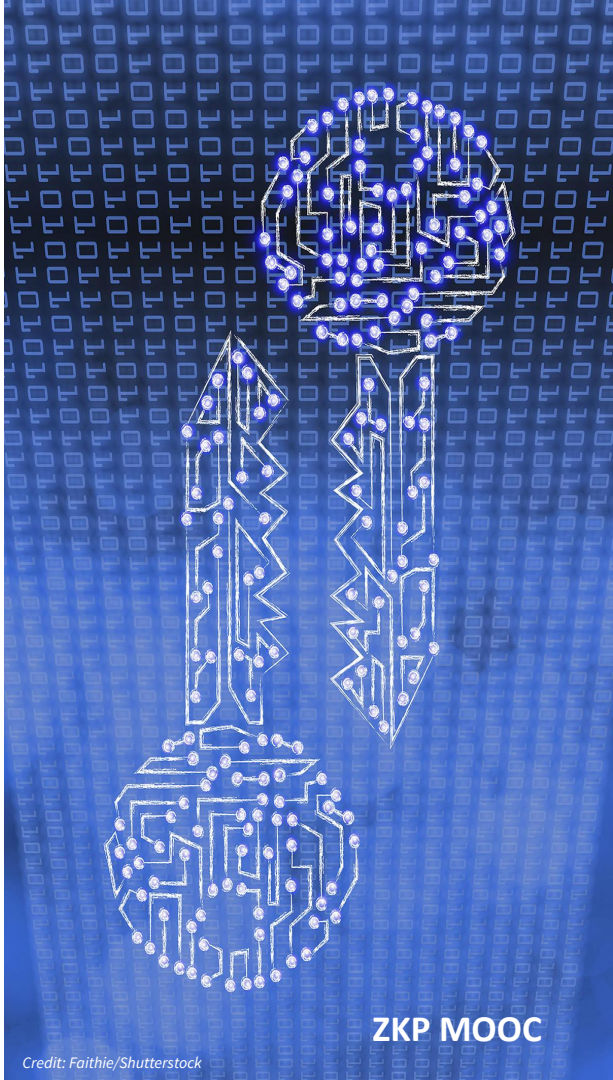
- **transparent setup**: no secret randomness in setup
- com_f is a single group element (independent of degree d)
- eval proof size for $f \in \mathbb{F}_p^{(\leq d)}[X]$ is $O(\log d)$ group elements
- eval verify time is $O(\log d)$ Prover time: $O(d)$

PCS have many applications

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



Proving properties of committed polynomials



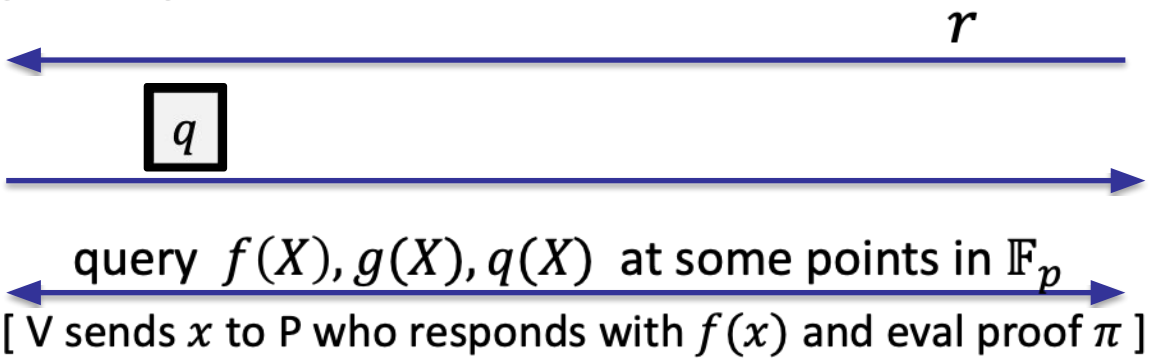
Proving properties of committed polynomials

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 an IOP:



Recall: polynomial equality testing

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

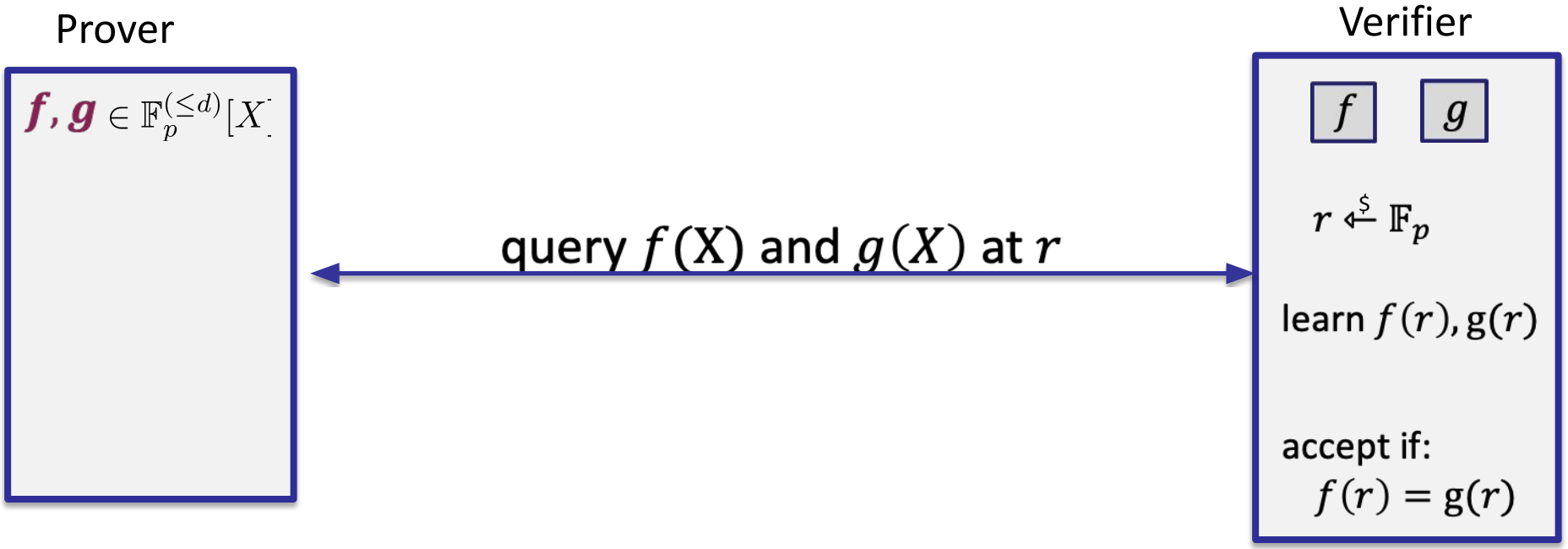
Let $f, g \in \mathbb{F}_p^{(\leq d)}[X]$.

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

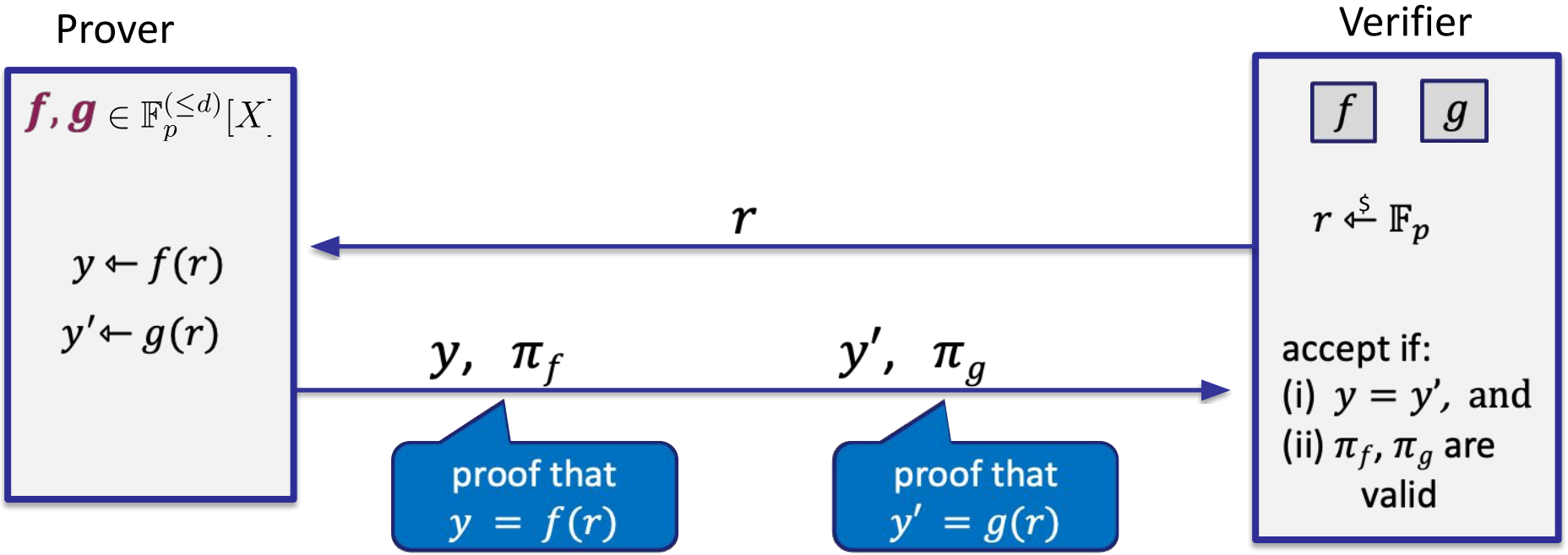
$f(r) - g(r) = 0 \quad \Rightarrow \quad f - g = 0$ w.h.p

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

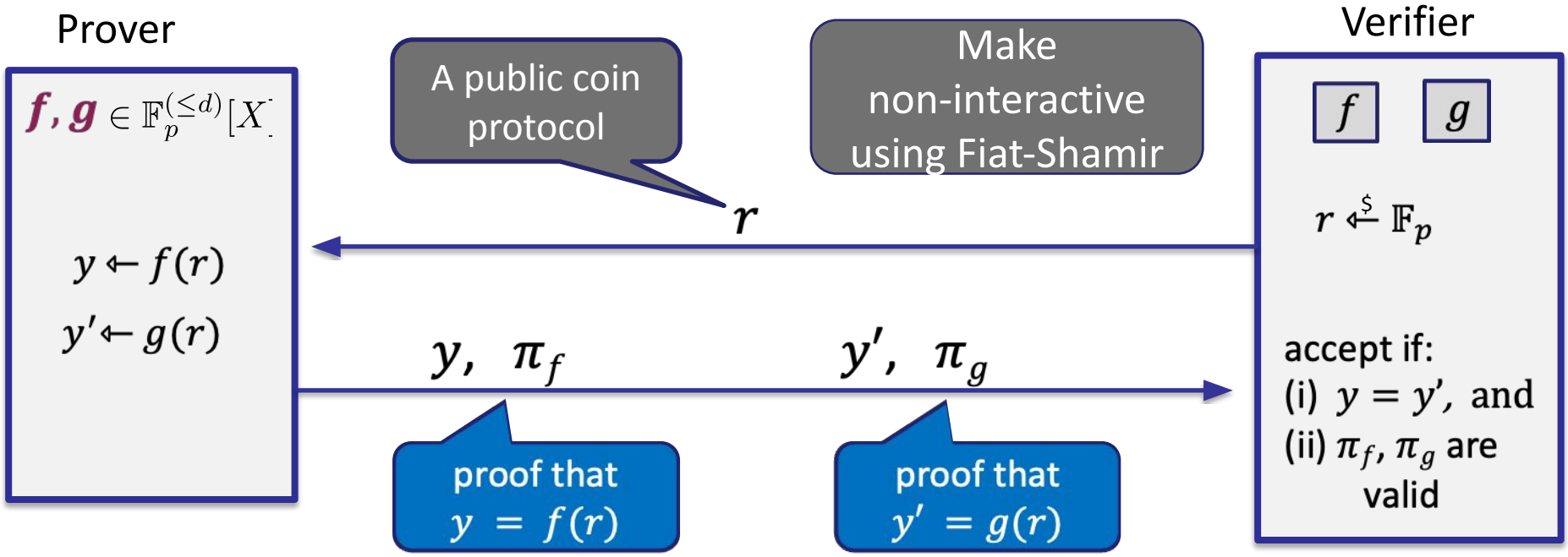
Review: the proof system as an IOP



Review: the compiled proof system



Review: the compiled proof system



Polynomial equality testing with KZG

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

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

But prover is needed to test equality of computed polynomials

- Example: verifier has f, g_1, g_2, 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 Ω be some subset of $\mathbb{F}_p$ of size k.

Let $f \in \mathbb{F}_p^{(\leq d)}[X]$ $\quad (d \geq k)$ $\quad$ 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 $\omega^k = 1$).

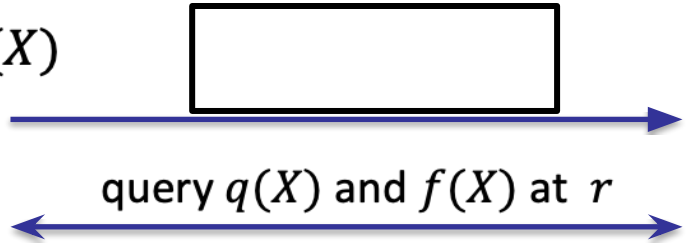
- if $\Omega = \{1, \omega, \omega^2, \ldots, \omega^{k-1}\} \subseteq \mathbb{F}_p$ then $Z_\Omega(X) = X^k - 1$

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

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

Prover P(f)

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



Verifier V(f)

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

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

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

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

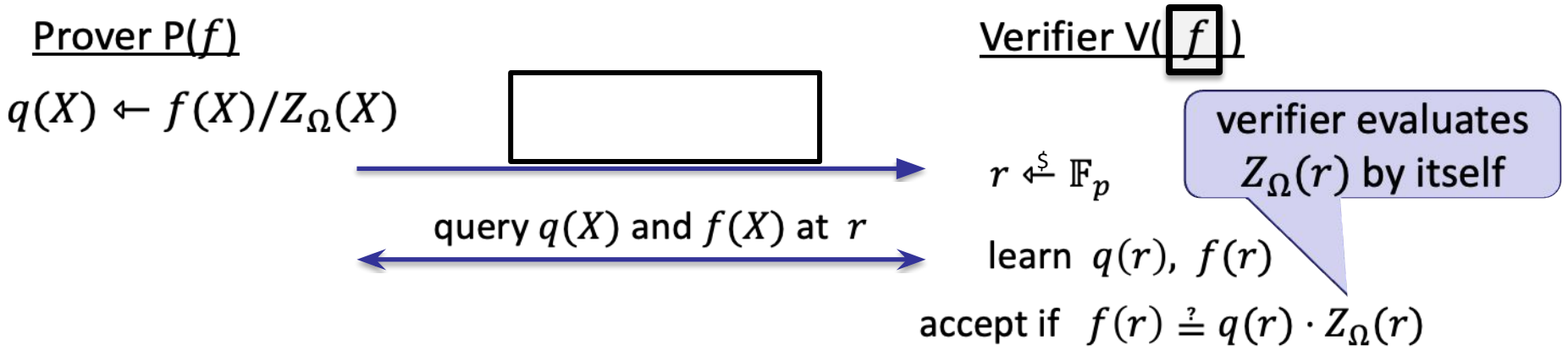
accept if $f(r) \stackrel{?}{=} q(r) \cdot Z_\Omega(r)$

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



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 Ω : $\prod_{a\in\Omega} f(a) = 1$

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

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

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 Ω : $\prod_{a\in\Omega} f(a) = 1$

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 Ω (unoptimized)

Prover P(f) — Verifier V(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 Ω

t q

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

query $t(X)$ at ω^{k-1}, r, ωr

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

query $q(X)$ at r, and $f(X)$ at ωr

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 Ω (unoptimized)

Prover P(f) — Verifier V(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 q

A public coin protocol

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

query $t(X)$ at ω^{k-1}, r, ωr

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

query $q(X)$ at r, and $f(X)$ at ωr

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(f , 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 f, g.

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

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

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

(4) Another useful gadget: permutation check

Prover P(f, g) — Verifier V(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: $\hat{f}(X) = \hat{g}(X) \iff g$ 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

accept or reject

[two commits, six evals]

[Lipton's trick, 1989]

(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 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 Ω

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

⇒ prover would need to manipulate polynomials of degree k^2

⇒ 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 $\left(W(a), f(a)\right)_{a\in\Omega}$ is a permutation of $\left(a, g(a)\right)_{a\in\Omega}$

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(f, g, W)

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

(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(f, g, 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 Ω

product check, sum check

permutation check

prescribed permutation check

Ok, I lied,one more gadget: range check

Prover P(f) Verifier V(f)

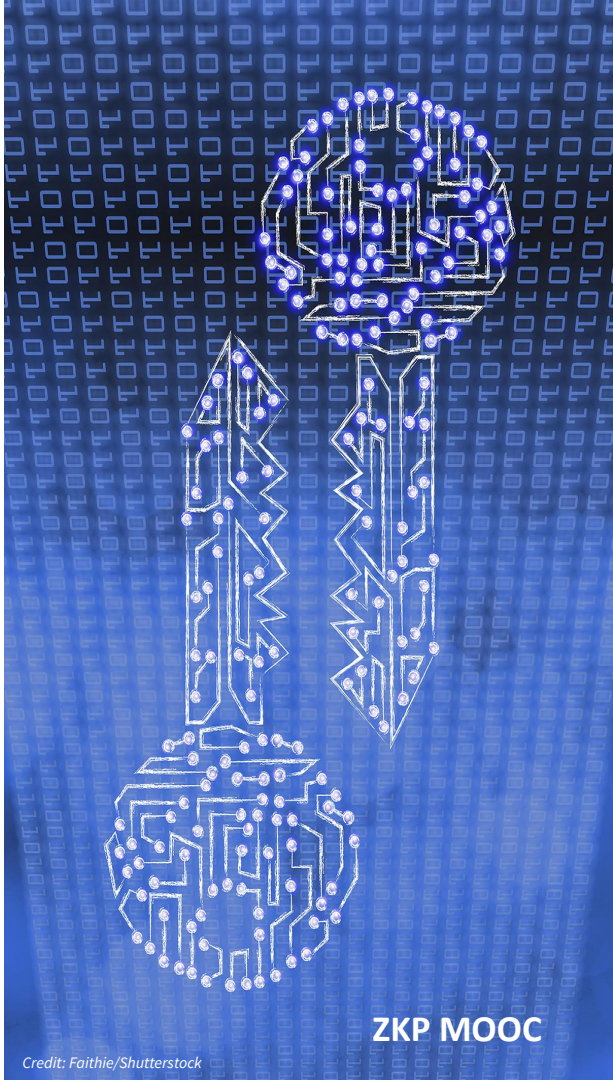
Goal: convince verifier that $f(a) \in \{0,1,2,\ldots,255\}$ for all $a \in \Omega$

Homework: design a proof system where the prover commits to an auxiliary polynomial of degree $\deg(f) + 256$

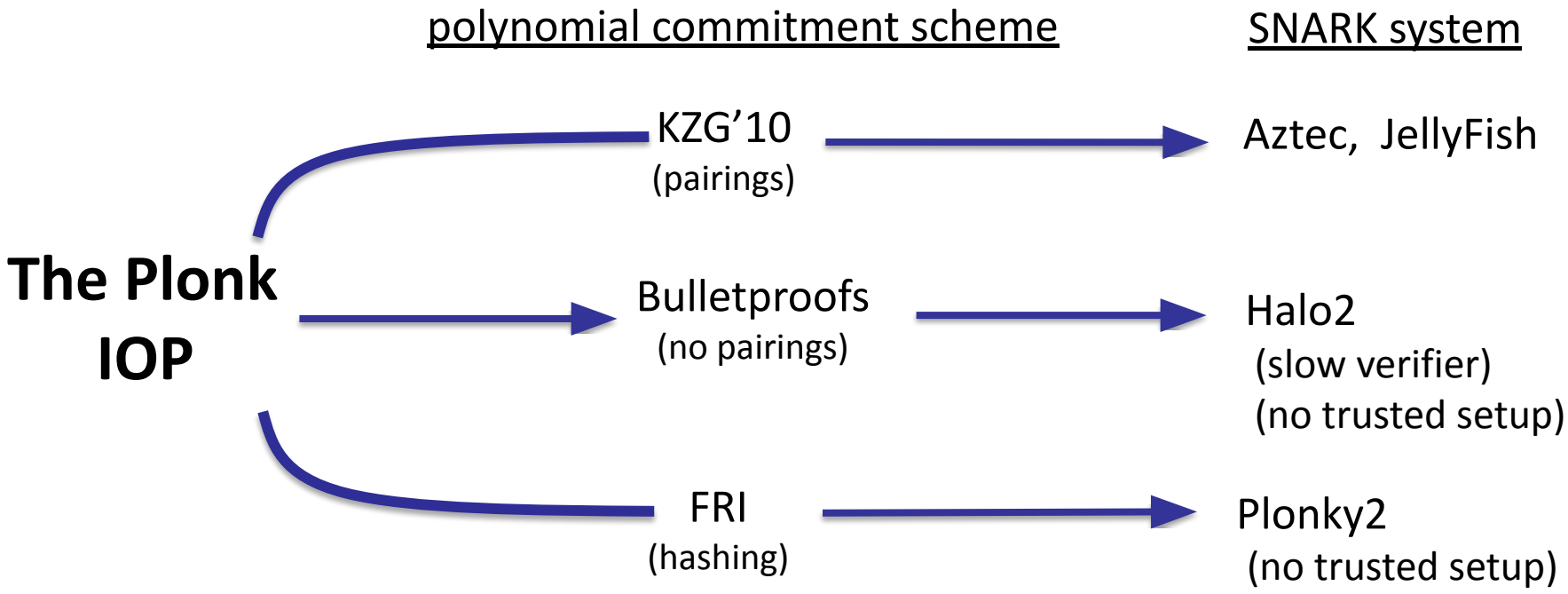
Hint: use a permutation check

The PLONK IOP for general circuits

eprint/2019/953

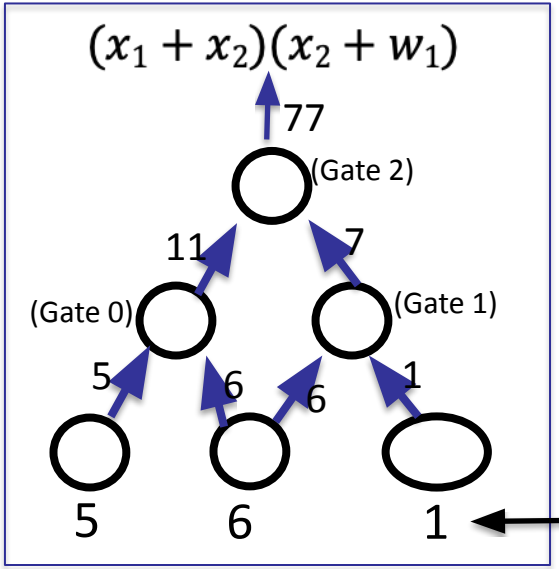


PLONK: widely used in practice



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

The plan:

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

that encodes the entire trace.

Let's see how …

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

Encoding the trace as a polynomial

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

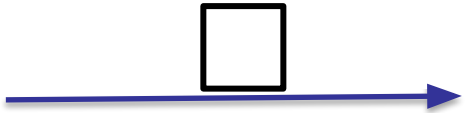
inputs:	$T(\omega^{-1}) = 5,$	$T(\omega^{-2}) = 6,$	$T(\omega^{-3}) = 1,$
gate 0:	$T(\omega^{0}) = 5,$	$T(\omega^{1}) = 6,$	$T(\omega^{2}) = 11,$
gate 1:	$T(\omega^{3}) = 6,$	$T(\omega^{4}) = 1,$	$T(\omega^{5}) = 7,$
gate 2:	$T(\omega^{6}) = 11,$	$T(\omega^{7}) = 7,$	$T(\omega^{8}) = 77$

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,	77

Step 2: proving validity of 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:

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

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 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 Ω_{inp} to prove that

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

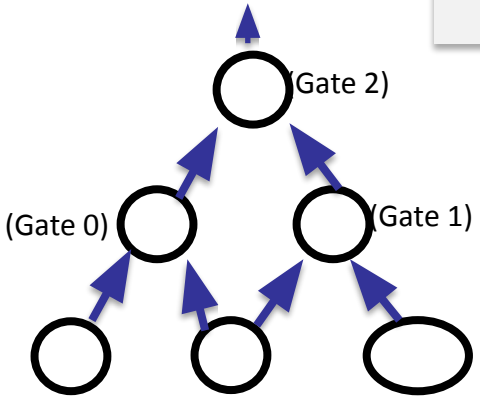
Proving (2): every gate is evaluated correctly

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

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

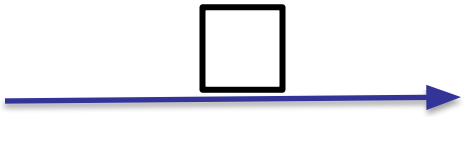
left input | right input | left input | right input | output

Proving (2): every gate is evaluated correctly

Setup(C) $\rightarrow$ pp:=S and vp:= (S)

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

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



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

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:

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

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**

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

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

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:

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

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: $\omega^{1},\ \omega^{3},\ \omega^{-2}$: **6, 1, 7**

2: $\omega^{2},\ \omega^{4},\ \omega^{-4}$: **11, 7, 77**

Proved using a prescribed permutation check

Define a polynomia[l] $\to \Omega$ that implements a rotation:

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

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

The complete Plonk Poly-IOP (and SNARK)

- Setup(C) $\rightarrow$ $pp := (S, W)$ and $vp := ($ S and W $)$ (untrusted)

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

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

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

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

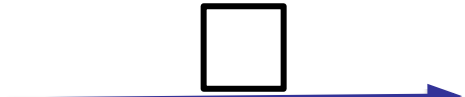
wires: (3) $T(y) - T(W(y)) = 0$ (using prescribed perm. check) $\forall\, y \in \Omega$

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

The complete Plonk Poly-IOP (and SNARK)

- Setup(C) $\rightarrow$ $pp := (S,W)$ and $vp := ($ S and W $)$

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

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

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