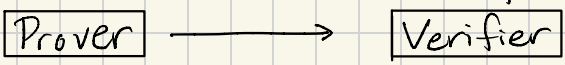
Lecture 13

- Poly IOP definition
- Fibonacci Example
- PLONK (Arithmetic Circuit) Example

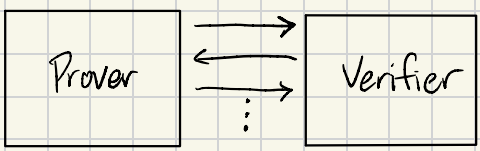
Roadmap Until Now: Info-Theoretic Proof Systems
→ secure against unbounded adversaries (no cryptographic assumptions)

NP: traditional model for proofs

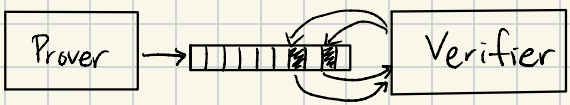


So far, we've seen two different extensions:

IP: add randomness + interaction



PCP: add randomness and oracle access to proof



Today: Polynomial Interactive Oracle Proof (PIOP): add randomness, interaction, and oracle access to polynomials

What is an oracle?
- abstract black box entity that answers queries of a certain type

ex) Random oracle for some $H \xleftarrow{\$} Funs[X, Y]$
Query: $x \in X$
Answer: $H(x) \in Y$

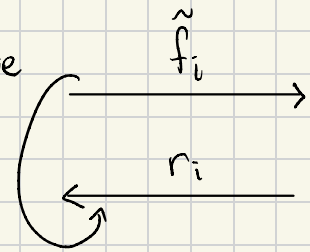
ex) PCP oracle for some π of length n
Query: $i \in [n]$
Answer: $\pi[i]$

ex) Polynomial oracle for some polynomial $f: X \to Y$
Query: $x \in X$
Answer: $f(x)$

A PIOP System for a language L is a tuple of efficient interactive algs (P, V) where the prover's messages are restricted to black-box polynomials $\in \mathbb{F}[X]$, which the verifier can query as oracles

$P(x, w)$ $\qquad$ $V(x)$

For round i, generate a set of black box polys $\tilde{f}_i$



can query $\tilde{f}_i$

sample randomness $\xleftarrow{\$} \{0,1\}^c$

↓

Output $\{0,1\}$

Completeness: $\forall x \in L$,
$\Pr[\langle P, V \rangle(x) = 1] \geq 1 - \text{negl}(\lambda)$

Soundness: $\forall x \notin L$, $\forall P^*$ that sends message polynomials whose deg $\leq \text{poly}(\lambda)$,
$\Pr[\langle P^*, V \rangle(x) = 1] \leq \text{negl}(\lambda)$

Why do we care about PIOPs?
↳ to generate SNARGs in practice, we combine information theoretic proof system with a cryptographic commitment

Info Theoretic System + Cryptographic Commitment = SNARG

PCP + Merkle Tree

too impractical to construct

PIOP + Polynomial Commitment

this lecture

Preliminaries

Define $\mathbb{F}$ to be a field of order p s.t.
- $|p| \approx 2^\lambda$
- $3 \cdot 2^\ell \mid p-1$ for $\ell \in \mathbb{N}$, $2^\ell \approx \text{poly}(\lambda)$

Since $\mathbb{F}^*$ is cyclic, $\exists$ a multiplicative subgroup $H \subseteq \mathbb{F}^*$ whose order is $n = 3 \cdot 2^\ell$. Furthermore, $\exists$ a generator $g \in \mathbb{F}^*$ s.t.

$$H = \{g^0, g^1, \ldots, g^{n-1}\}$$

We will refer to H as our evaluation domain.

Vanishing Polynomial: the roots of $X^n - 1 \in \mathbb{F}[X]$ are exactly H

Fact: $(x^n - 1) \mid f(x) \iff f(x)$ "vanishes on H"

Inverse FFT Transform: interpolates coeffs of f from evals over H in $O(n \log n)$ time

$$FFT^{-1}(f(1), f(g), \ldots, f(g^{n-1})) \to f$$

Fibonacci Example

Fibonacci Sequence

$t_0 = 0,\ t_1 = 1,\ \forall j \geq 2,\ t_j := t_{j-2} + t_{j-1}$

Index	0	1	2	3	4	5	6	...
Term	0	1	1	2	3	5	8	...

$R_{fib} := \{ (j, t_j); (t_0, t_1, \ldots t_{j-1}) : (t_0, \ldots t_j)$ are first j terms of Fib sequence $\}$

We will describe a PIOP for this trivial relation where the verifier does a constant # of queries and a small number of arithmetic ops

Intuition

- Prover could commit to its witness as a polynomial $f \in \mathbb{F}^{<n}[X]$ s.t. $\forall i \leq j, f(g^i) = t_i$ (assume $j = n-1$ for simplicity)

x	g^0	g^1	- - -	g^{n-1}
f(x)	t_0	t_1	- - -	t_j

← these are simply all elements in H!

→ we encode the witness as evaluations over H. Then, the prover could send f oracle to the verifier. The verifier could query $f(g^{n-1})$ to get the j^{th} value

↳ But how can the verifier be sure that f(x) actually encodes the correct values of the Fibonacci sequence???

Polynomial Property Testing

We want to test whether f encodes the Fibonacci sequence

1) The verifier can query $f(g^0) \stackrel{?}{=} 0$, $f(g^1) \stackrel{?}{=} 1$

2) We now want to check if $\forall h = g^i \in H \setminus \{g^0, g^1\}$:

$$t_{i-2} + t_{i-1} = t_i$$
$$f(g^{i-2}) + f(g^{i-1}) = f(g^i)$$
$$f(g^{-2}h) + f(g^{-1}h) = f(h)$$

Rearranging:

$$f(g^{-2}h) + f(g^{-1}h) - f(h) = 0$$

Another way of viewing this is that we want all $h \in H \setminus \{g^0, g^1\}$ to be zeroes of the polynomial $f(g^{-2}x) + f(g^{-1}x) - f(x)$

$\iff$ $F(x) := (f(g^{-2}x) + f(g^{-1}x) - f(x))(x - g^0)(x - g^1)$ vanishes on H

Using our earlier fact that

$F(x)$ vanishes on $H \iff x^n - 1 \mid F(x)$

The prover can derive a quotient poly $q(x) := \frac{F(x)}{x^n - 1}$ and send to verifier

Notice if $F(x) \neq (x^n - 1)\, q(x)$, then $F(x) - (x^n - 1) q(x) \neq 0$

Thus, the verifier can check

$$F(x) \stackrel{?}{=} (x^n - 1)\, q(x)$$

by querying q and f at a random point $\alpha \xleftarrow{\$} \mathbb{F}$

$$F(\alpha) \stackrel{?}{=} (\alpha^n - 1)\, q(\alpha)$$
$$(f(g^{-2}\alpha) + f(g^{-1}\alpha) - f(\alpha))(\alpha - g^0)(\alpha - g^1) \stackrel{?}{=} (\alpha^n - 1)\, q(\alpha)$$

P

- interpolate $f \leftarrow FFT^{-1}((f(g^i) = t_i,\ \forall i \le j))$
- define $F(x) := (f(g^{-2}x) + f(g^{-1}x) - f(x))(x - g^0)(x - g^1)$
- compute quotient poly $q(x) := F(x)/(x^n - 1) \in \mathbb{F}[X]$
- send f, q to verifier

V(j, t_j)

- Query f at g^0, g^1, g^j to check
  $$f(g^0), f(g^1) = 1, f(g^j) = t_j$$
- Sample $\alpha \xleftarrow{\$} \mathbb{F}$, query f at $g^{-2}\alpha, g^{-1}\alpha, \alpha$ to compute
  $$F(\alpha) = (f(g^{-2}\alpha) + f(g^{-1}\alpha) - f(\alpha))(\alpha - g^0)(\alpha - g^1)$$
- Query q at α to check $(\alpha^n - 1)q(\alpha) = F(\alpha)$
- Output accept if checks pass

Completeness

- $x^n - 1 \mid F(x)$ iff H is a set of roots of F
- We constructed $F(x)$ and f s.t. $\forall h \in H \setminus \{g^0, g^1\}$,
  $$f(g^{-2}h) + f(g^{-1}h) - f(h) = 0$$
- For $h \in \{g^0, g^1\}$
  $$(x - g^0)(x - g^1) = 0$$

Thus, F vanishes on H

Soundness

Say a malicious prover P^* claims that $t_j = c$ where c is not the j^{th} Fibonacci number.
They send polys $f, q \in \mathbb{F}[X]$ s.t. $\deg(f), \deg(q) \le \text{poly}(\lambda)$

Let us denote the event E_0 as the event where:
$$f(g^0) = 0,\ f(g^1) = 1,\ f(g^j) = c$$
and denote E_1 as the event where $(x^n - 1)q(X) = F(x)$

$$\Pr[V \text{ accepts}] = \Pr[V_{acc} \mid E_0] \cdot \Pr[E_0] + \Pr[V_{acc} \mid \neg E_0] \Pr[\neg E_0]$$
$$\le \Pr[V_{acc} \mid E_0] + 0$$

$= \Pr[V_{acc} \mid E_1 \wedge E_0]\Pr[E_1 \mid E_0] + \Pr[V_{acc} \mid E_0 \wedge \neg E_1]\Pr[\neg E_1 \mid E_0]$
$\leq \Pr[E_1 \mid E_0] + \Pr[V_{acc} \mid E_0 \wedge \neg E_1]$

$\underline{\Pr[V_{acc} \mid E_0 \wedge \neg E_1]}$:

If $(x^n-1)\,q(x) \neq F(x)$, then $\overbrace{(x^n-1)q(x) - F(x)}^{Q(x)} \neq 0$. Thus,

$\Pr[(\alpha^n-1)q(\alpha) - F(\alpha) = 0 \mid \alpha \xleftarrow{\$} H] \leq \frac{\deg(Q)}{|F|} = \frac{poly(\lambda)}{|F|} \leq negl(\lambda)$

Q can have at most $\deg(Q)$ roots over F

Thus, $\Pr[V_{acc} \mid E_0 \wedge \neg E_1] \leq negl(\lambda)$

$\underline{\Pr[E_1 \mid E_0]}$:

We will argue $E_0 \wedge E_1$ cannot occur $\Rightarrow \Pr[E_1 \mid E_0] = 0$

Lemma: If $\exists f, q \in F[X]$, s.t. $(x^n-1)q(x) = F(x)$ and $f(g^0) = 0$, $f(g^1) = 1$, $f(g^j) = c$, then $c = t_j$.

$(x^n-1)q(x) = F(x) \iff F$ vanishes on H

$\iff \forall h \in H,\ (f(g^{-2}h) + f(g^{-1}h) - f(h))(h-g^0)(h-g^1) = 0$

$f(g^0) = c$, $f(g^1) = 1$, $f(g^j) = c$

By induction on indices, $t_j = c$

By contrapositive of lemma, if $c \neq t_j$, then $\nexists f, g$ s.t. $E_0 \wedge E_1$ occurs. Thus, $\Pr[E_1 \mid E_0] = 0$

Therefore $\Pr[V_{acc}] \leq 0 + negl(\lambda) \leq negl(\lambda)$

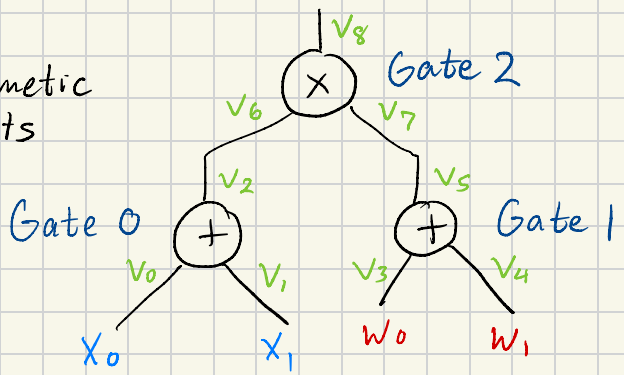
PLONK (GWC 19)

- SNARK for Circuit-SAT
- clever encoding of arithmetic circuits

Arithmetic Circuits

- an arithmetic circuit is a directed acyclic graph where each node is an addition or multiplication gate w/ input-arity 2 and output arity 1. The edges (wires) take on values from a specified field $\mathbb{F}$.
- these wires must satisfy the constraints enforced by the gates

- this simple arithmetic circuit w/ inputs $(x,w) \in \mathbb{F}^4$ calculates $x_1 x_2 (w_1 + w_2)$



We want to create a PIOP for the following index-relation

$$R_C := \{ (x; w) \mid C(x, w) = 0 \}$$

* recall from last lecture that when proving index-relations, we allow for an offline preprocessing phase to help with proof → for AC, offline preprocessing involves deriving and committing to polynomials that enforce constraints specified by the particular circuit

Let's use the above circuit as an example. We call the vector $V := \{v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ the extended witness. For v to be a valid assignment of the wires, the following 2 checks must hold:

1) Gate Checks: $v_0 + v_1 \stackrel{?}{=} v_2$

$v_3 + v_4 \stackrel{?}{=} v_5$

$v_6 \times v_7 = v_8$

2) Copy Checks: $(v_0, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8) \stackrel{?}{=}$

$(v_0, v_1, v_6, v_3, v_4, v_7, v_2, v_5, v_8)$

Let's define 2 vectors based on C to help w/ these checks:

- Gate selector S: $(0, 0, 1)$
  - ↳ $s_i = 0$ is Gate i is an addition gate and 1 if Gate i is a multiplication gate
- Permutation vector Π: $(0, 1, 6, 3, 4, 7, 2, 5, 8)$

  $(0, 1, 2, 3, 4, 5, 6, 7, 8)$

Notice checks 1) and 2) are equivalent to checking:

1) $\forall i < 3, \quad j = 3i$

$$(1 - S_i)(V_j + V_{j+1}) + S_i (V_j \cdot V_{j+1}) = V_{j+2}$$

2) $\forall j < 9, \quad V_j \stackrel{?}{=} V_{\Pi_j}$

We will now translate these checks into statements about polynomials!

To encode circuit C as a polynomial, interpolate polynomials $S(x)$ and $\Pi(x)$ (assuming $|H| = 9$, $H' \subseteq H$, $H' = \{g^0, g^3, g^6\}$) s.t. $S(g^{3i}) = S_i$ and $\Pi(g^i) = \Pi_i$

x	g^0	g^3	g^6
$s(x)$	0	0	1

x	g^0	g^1	g^2	g^3	g^4	g^5	g^6	g^7	g^8
$\pi(x)$	g^0	g^1	g^6	g^3	g^4	g^7	g^2	g^5	g^8

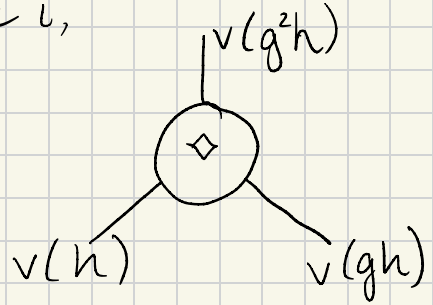
These polynomials will be output by V in preprocessing

The prover can interpolate a polynomial $v(x)$ s.t.

x	g^0	g^1	- - -	g^8
$v(x)$	v_0	v_1	- - -	v^8

The prover sends $V(x)$ to the online verifier, who has oracle access to $s(x)$ and $\pi(x)$

Note that we have defined $s(x)$, $V(x)$ such that for each gate i, let $h = g^i$:



◇ = + if $s(h)=0$
× if $s(h)=1$

To perform gate check, the verifier would like to check $\forall h \in H'$:

$$(1 - s(h))(v(h) + v(gh)) + s(h)(v(h)\,v(gh)) = w(g^2 h)$$

This is equivalent to checking if the polynomial

$$(1 - s(x))(v(x) + v(gx)) + s(x)(v(x)\,v(gx)) - v(g^2 x)$$

vanishes on $H' \Rightarrow$ can use same quotient trick from Fibonacci proof

Checking copy constraints is more complicated. We must efficiently test if

$$V(h) = V(\pi(h)) \text{ for all } h \in H$$

Using our quotient strategy to test if

$$V(x) = V(\pi(x))$$

vanishes on H would require the prover to compute a quotient $q(x)$ whose degree $\approx n^2$ (quadratic)

↳ turns out that with clever randomized tests, we can check this without a quadratic blow-up!