Lecture 12: Linear PCPs

MIT - 6.893

Fall 2020

Henry Corrigan-Gibbs

Plan

* Recap: MPC apps
* Proofs and ZK
* Linear PCPs

Logistics

* HW 4 out now
  due 10/30 @ 5pm
* OH today 3-4:30pm

Two comments on homework

1. On many T/F questions...

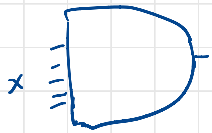
"If P = NP, then ______ exists."

Subtext: Does ______ require computational assumptions?

If P=NP then ~~PRGs~~, ~~PKE~~, ~~DDH~~, ...

Many ways to see this. One way is that if P=NP then CIRCUIT-SAT has a poly-time alg.

Can use ckt SAT to solve Dlog or break any other OWF:

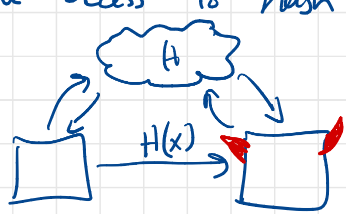


$$\begin{cases} 1 & \text{if } y = g^x \in \mathbb{G} \\ 0 & \text{o.w.} \end{cases}$$

But, can't use ckt SAT to invert a random oracle...

2. On random-oracle model

In ROM, all parties have access to hash fn H (modeled as R.O.)



Adv can test whether $x' \stackrel{?}{=} x$ by querying H

↳ formally, can't simulate.

Recap: MPC Applications

Estonia: Students & Taxes

3PC to overcome regulatory barriers to data sharing

Genomics: 2PC to compute genome association data for rare genetic diseases

MPC overcomes data privacy concerns

Google: 2PC for business data sharing.

We did not discuss private aggregation (essentially MPC w/o *)

↳ Lots of apps we will cover soon...

Zero-knowledge Proofs

One of the most beautiful concepts in all of CS.

A ZK proof is convincing but not revealing.

- e.g. V is convinced that C is SAT but "learns nothing" about SAT assignment.

- e.g. V convinced that $N \in \mathbb{Z}$ is product of exactly two primes w/o learning what they are.

Most standard crypto classes will cover theory of ZK proofs.

e.g. ZK proof systems for arbitrary NP languages

Here, we will focus on concretely efficient modern ZK proofs + applications.

We will not cover so many beautiful things that are worth knowing...

IP = PSPACE, GKR, GI protocol, ...

What do we mean by "proof"?

Goal of Proof: Convince someone of something.
"verifier" "statement"

Examples: "$N \in \mathbb{Z}$ is the product of exactly two primes"

"The Pythagorean Thm is true."

"C is a ckt without a satisfying assignment"
$C \in$ CIRCUIT-UNSAT

⋮

For this class, we will only consider statements of the form:

$$\boxed{\text{"Arithmetic circuit } C \text{ is satisfiable." (over finite field } \mathbb{F})}$$

Recall: An arithmetic circuit $C: \mathbb{F}^n \to \mathbb{F}$ is like a Boolean ckt w/ + and $\times$ gates in $\mathbb{F}$.

An ath ckt $C: \mathbb{F}^n \to \mathbb{F}$ is satisfiable if $\exists\, x \in \mathbb{F}^n$ s.t. $C(x) = 0_{\mathbb{F}}$.

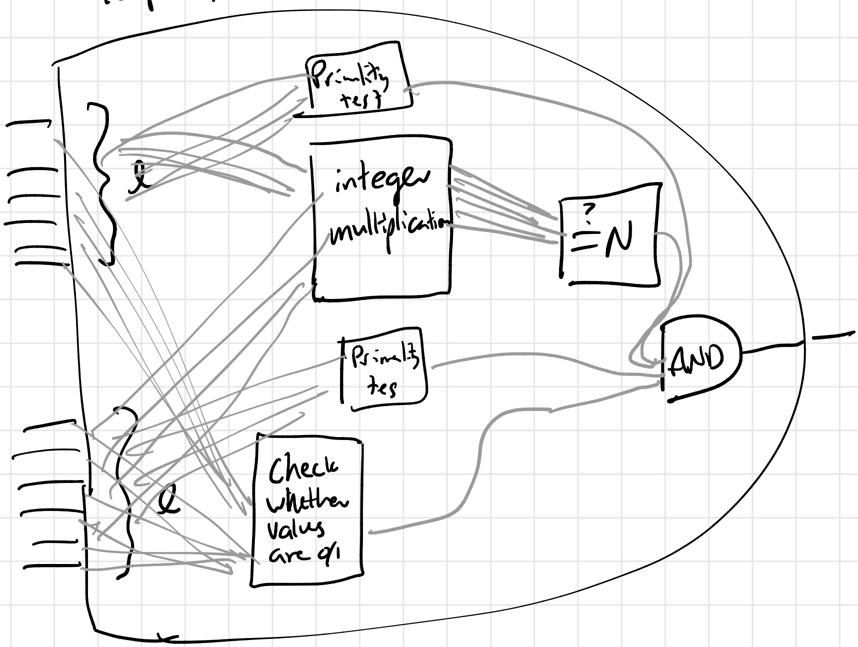
As we saw earlier: (informally) if f is a poly-time computable fn, then there's a small (poly-size) arithmetic ckt that computes f.

↳ Proof systems that can handle statements of this form can capture all NP languages.

Example

"$N \in \mathbb{Z}$ is the product of exactly two ℓ-bit primes"

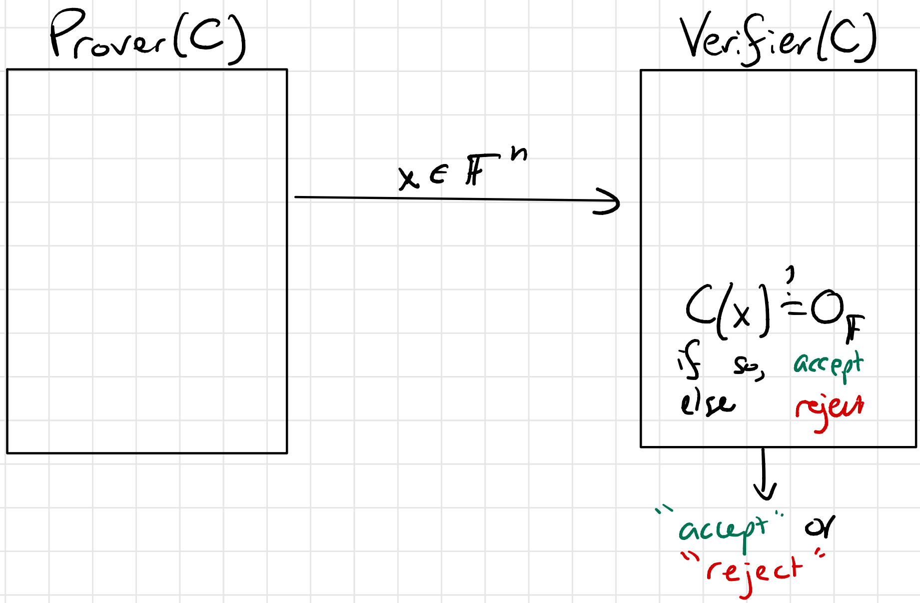
$C_{\text{twoprimes}, N}$



Circuit

$$C(p_1, \ldots, p_\ell, q_1, \ldots, q_\ell) = \begin{cases} \text{Check all inputs} \in \{0,1\}_{\mathbb{F}} \\ \text{Compute } p \leftarrow \sum_{i=1}^{\ell} 2^{i-1} p_i \\ \qquad\qquad q \leftarrow \sum_{i=1}^{\ell} 2^{i-1} q_i \\ \qquad\qquad N' = p \cdot q \in \mathbb{Z} \\ \text{Output } 0 \text{ if } N' = N \end{cases}$$

If you want to convince your friend that ckt C is SAT, you just send the sat input.



Potential problem:
Verifier learns satisfying input!

ZK Proof Systems (informally)

Interaction b/w Prover P and verifier V.
Let $\langle P(C), V(C) \rangle$

Properties:

1. Completeness: $\forall$ sat ckt C

$$\Pr[\langle P(C), V(C) \rangle = \text{"accept"}] \geq 2/3.$$

2. Soundness: $\forall$ unsat ckt C $\forall P^*$

$$\Pr[\langle P^*, V(C) \rangle = \text{"accept"}] \leq 1/3.$$

3. Zero knowledge

V "learns nothing" from P except that C is sat.

Formalize w/ simulation

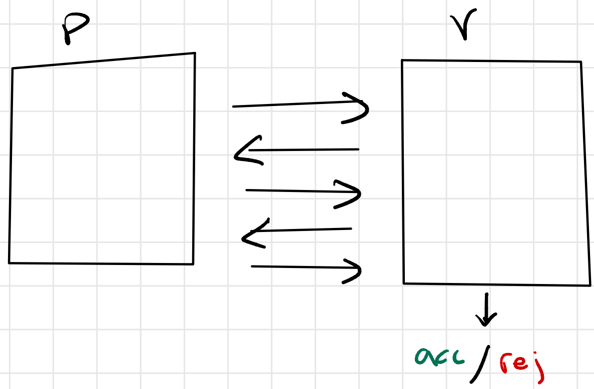
*** In particular, V does "not learn anything" about satisfying input to C.

Notice: V is randomized!

ZK Proofs

To prove NP statements in ZK, generally need more complicated $P \leftrightarrow V$ interaction...

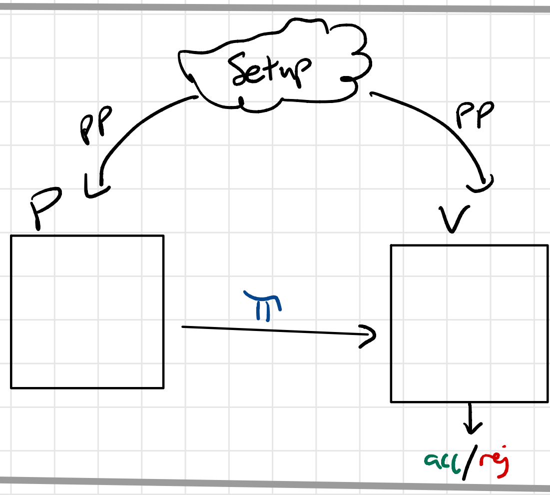
Interaction



"Trusted" Setup

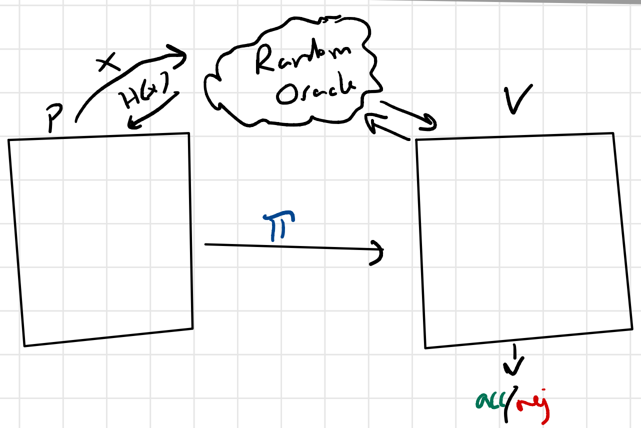
NIZK, SNARK, ...

→ Will see one example of this next class.



Random Oracle

CS Proofs, STARK, ...



Plan for next three classes

Will try to avoid overlap w/ 6.875, 6.857...

Today: A useful building block for modern ZK proofs ... what implementations use today.

Next week: Use the tool to construct...

Succinct ZK Proofs

Idea: Convince verifier that ckt C has a satisfying input where V runs in less time than needed to evaluate ckt.

ZK Proofs on Secret-Shared Data

Idea: P convinces set of parties that they hold shares of satisfying assignment to ckt C.

↳ We'll see applications

Linear PCPs

IKO07, BCIOP13, ...

* A building block to construct ZK proofs.
* Once you have a good LPCP, can "compile it" into various types of ZK proof systems.
  → Enormously fruitful strategy used of late.

In a normal proof interaction:

1) P sends π to V.
2) V reads π.
3) V accepts / rejects

In a LPCP V cannot explicitly read the proof.

→ V only gets access to proof via "linear queries"

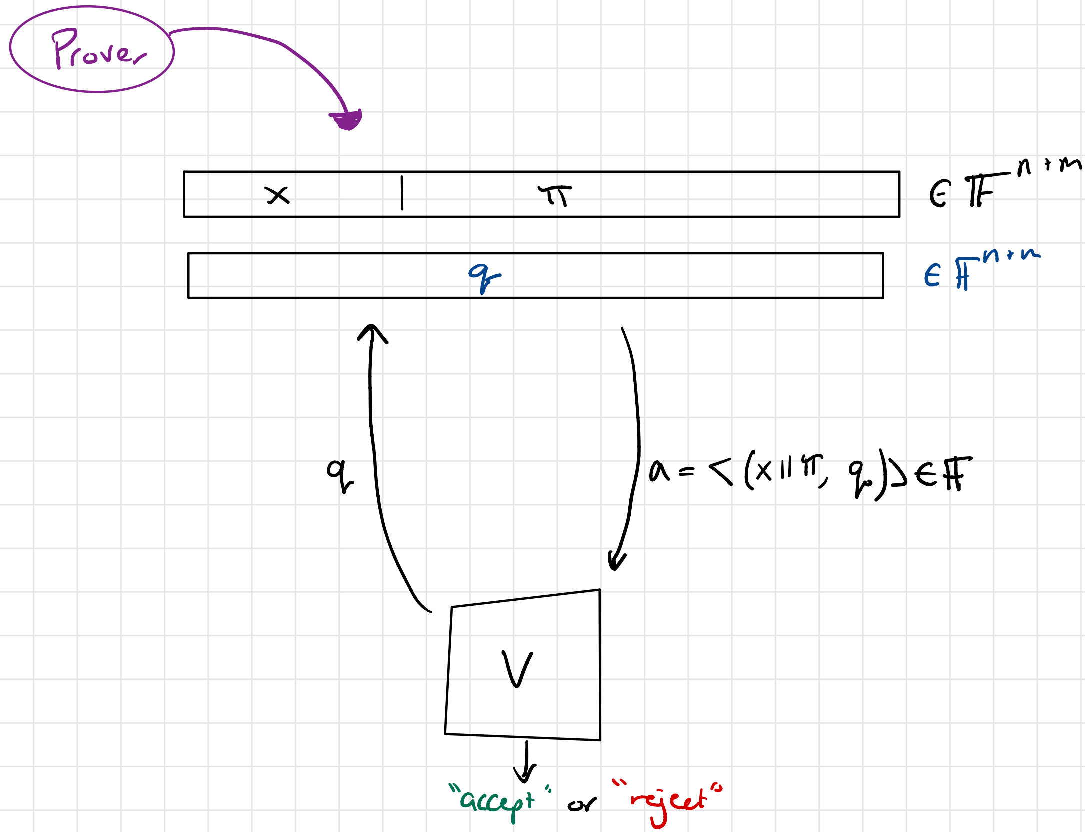
1) P outputs π
2) V makes $O(1)$ linear queries to π
3) V accepts / rejects.

Linear PCPs

IKO07, BCIOP13, ...

- Prover outputs claimed sat input $x \in \mathbb{F}^n$, extra stuff $\pi \in \mathbb{F}^m$
- Verifier gets to make "linear queries" to $(x \| \pi)$

Interaction b/w Prover and Verifier is...



Linear PCPs

In practice, we also care about time to construct π

1. Completeness.

If $C(x)=0$ then $\exists \pi$ s.t.

$$\Pr\left[V^{\langle \cdot, x \| \pi \rangle}() = \text{"accept"}\right] \geq 2/3.$$

2. Soundness.

If C is UNSAT then $\forall (x^*, \pi^*)$

$$\Pr\left[V^{\langle \cdot, x^* \| \pi^* \rangle}() = \text{"accept"}\right] \leq 1/3.$$

3. Honest Verifier Zero Knowledge.

$\exists$ simulator Sim s.t.

$$\left\{ \begin{array}{c} V\text{'s view in} \\ \text{interaction w/} \\ \text{proof oracle} \end{array} \right\} \overset{s}{\approx} \left\{ \text{Sim}() \right\}$$

How you construct linear PCPs is not so important.

Key thing to remember: [GGPR13, ...

$\boxed{\begin{array}{l}\text{If } C \text{ is a ckt over } \mathbb{F} \text{ of size } S, \text{ then there is a linear PCP for } C \text{ in which:}\\ \quad * \; V \text{ makes 4 queries}\\ \quad * \text{ proof has size } O(S). \\ \qquad\qquad\qquad (|\mathbb{F}| \gg S)\end{array}}$

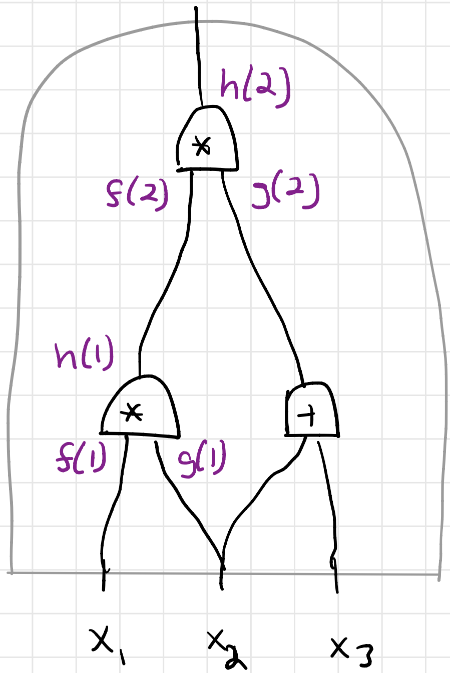
Why this is surprising:

Verifier in linear PCP gets only 4 field elements worth of info about input x and proof π.

And yet, V is able to tell "good" xs from bad ones w.h.p.

If we have time...

Construction of LPCP



1. Evaluate C on SAT input.

2. Define polynomials f, g, h s.t.

   $f(i)$ = value on LEFT input to ith mul gate

   $g(i)$ = value on RIGHT input to ith mul gate

   $h = f \cdot g$

3. Proof is coeffs of (f, g, h)

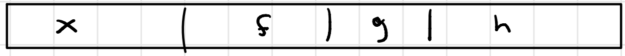
To check proof:

$\left.\begin{array}{l} * \ f, g \text{ are consistent with inputs} \\ * \text{ output (here } h(2)) \text{ is } 0 \in \mathbb{F} \\ * \text{ internal + gates and * constant gates} \end{array}\right\}$ One linear query

$* \ f(r) * g(r) = h(r)$ at random $r \in \mathbb{F}$ $\}$ Three linear queries

→ (To get ZK, set $f(0)$, $g(0)$ to random values)

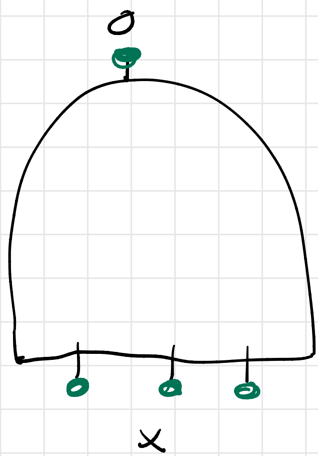
Linear Checks



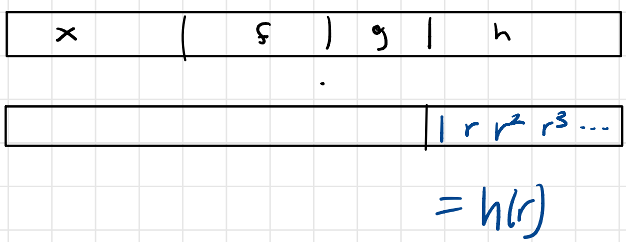
$$f(1) - x_1 = 0$$
$$f(2) - x_2 = 0$$
$$g(1) - x_2 = 0$$
$$g(2) - (x_2 + x_3) = 0$$
$$h(2) = 0$$

Idea: Take a random linear combination of these equations and check that it is $= 0$.

Together, these checks enforce the boundary conditions



Mul Checks



Each poly eval is one linear query.

Properties

Completeness: By construction.

Soundness: C is UNSAT.

$\rightarrow$ For any $x \in \mathbb{F}^n$, $C(x) \neq 0$.

Either $h(\ell) \neq 0$ or some $+$ gate computed incorrectly

$\hookrightarrow$ lin check fails

$\exists$ some i s.t. $f(i) \cdot g(i) \neq h(i)$

Then $f \cdot g \neq h \Rightarrow$ often $f(r) \cdot g(r) \neq h(r)$

$\hookrightarrow$ Mul check fails
By Schwartz-Zippel

HVZK: If $f(0), g(0)$ chosen at random, then if $r \in \{1, \dots, |C|\}$, query answers are just random elms of $|\mathbb{F}|$ (or zero).

Big Picture

Linear PCPs: Strange type of proof in which V gets restricted access to proof.

→ We will see two nice applications in next two lectures.