Lecture 8: IP/ZK

4/30/20

Plan

Interactive proofs
Zero-Knowledge
- what is it/why is it useful
- how do we define it?
- Example: HAMCYCLE

One of the most beautiful ideas in CS!

Logistics

- HW2 due Monday

Today we enter the 3rd "unit" of the course:

Unit 1 (lec 1-3): Foundations
Unit 2 (lec 4-7): Cryptanalysis & ECC
Unit 3 (lec 8-11): Zero-Knowledge

Proofs

Goal of a proof: Convince $\underbrace{\text{Someone}}_{\text{"Verifier"}}$ that $\underbrace{\text{Something}}_{\text{"Statement"}}$ is true.

How to formalize Statement?

Notion from Complexity theory: A Language is a set of strings $L \subseteq \{0,1\}^*$

A statement takes the form $x \in L$

Some familiar examples:

"N is the product of exactly 2 primes"

$$N \in \{pq \mid p,q \text{ prime}\}$$

"The Pythagorean Thm is true"

$$\text{PYTHM} \in \{\text{true statements in some formal system}\}$$

"ϕ is an unsatisfiable SAT formula"

$$\phi \in \{\text{set of unsatisfiable SAT instances}\}$$

A.K.A. $\phi \in \text{coSAT}$

Conventional Proofs and NP

Conventional proof:

$$\text{Prover}(x) \xrightarrow{\pi} \text{Verifier}(x) \downarrow \text{"accept" or "reject"}$$

- π might be hard to find
- π should be "easy" to check (polynomial time, deterministic verifier)

Definition of complexity class NP: L is in NP if statement $x \in L$ can be proven with a conventional pf.

Completeness: $x \in L \to V$ always accepts π
Soundness: $x \notin L \to V$ never accepts π

Formally: There exists an efficient algorithm $M(\cdot,\cdot)$ s.t.
$x \in L \leftrightarrow \exists w \in \{0,1\}^{\text{poly}(|x|)}$ s.t. $M(x,w)=1$.

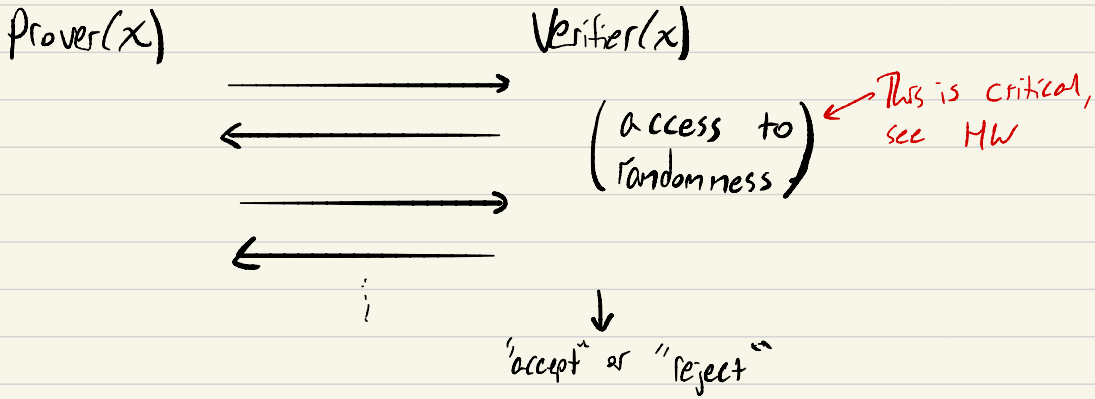
Example: to prove that $\phi \in SAT$, P sends satisfying assignment to V

More generally, P sends $\pi \leftarrow w$ and V checks $M(x,w)=1$.

Interactive Proofs

In real life, people hold interviews, press conferences, depositions, etc. to get more information than they could from reading/listening to a prepared statement. Can we do the same for proofs?

Interactive Proofs:
- P & V allowed to interact
- V allowed to use randomness



How do completeness and soundness change now that V is randomized?

denotes protocol b/w P, V where both get input x

Can increase to $1-\text{negl}(\lambda)$

Completeness: $\forall x \in L \quad \Pr[\langle P, V\rangle(x) = \text{"accept"}] \geq 2/3$

Soundness: $\forall x \notin L, \forall P^* \quad \Pr[\langle P^*, V\rangle(x) = \text{"accept"}] \leq 1/3$

P is the honest prover algorithm. Soundness should hold for <u>any</u> malicious prover, not just the honest one.

Can decrease to $\text{negl}(\lambda)$ w/repetition

What do we get from interaction?

1) IP captures a much broader class of problems than NP. It turns out IP = PSPACE.
(take a complexity class to learn more)

2) Even for NP statements, interaction can allow proving a statement with communication less than $|\pi|$.
(we'll discuss this more in a couple weeks)

---

3) Interaction enables a surprising new property: Zero Knowledge!

ZK informally: V "learns nothing" from interaction with P, except that $x \in L$

Not clear what "learns nothing" means, but let's see a couple examples before formalizing it.

ex. Given ϕ, prove I know $\phi \in SAT$ without revealing the SAT assignment

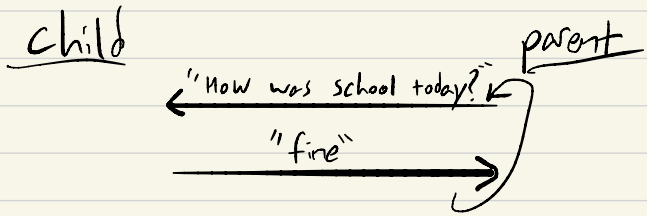
ex. Prove x is the correct output of some algorithm without revealing my secret inputs to the algorithm

ZK is used to define security in many protocols where we want to prove that "nothing leaks."

What does it mean to "learn nothing"?

Examples of interactions where you learn nothing:

ex1.



ex2. Spokesperson to reporter: "can neither confirm nor deny"

parent/reporter "learn nothing" in these examples because they could have known the response before asking.

Intuition: If V can easily write down a transcript of its interaction with P, then V hasn't learned anything useful from P.

Formalizing Zero Knowledge

Zero Knowledge: $\forall$ efficient V^*, $\exists$ efficient Sim s.t. $\forall x \in L$,

$$\{\mathrm{View}_{V^*}[\langle P, V^* \rangle(x)]\} \approx_c \{\mathrm{Sim}(x)\}$$

Needs to work for any (potentially malicious) verifier V^*, not just the honest verifier V.
If we wrote the definition with V instead of V^* this is called "Honest Verifier ZK" or HVZK.

Different flavors of ZK

perfect	$=$
Statistical	$\approx_s$
Computational	$\approx_c$

$\mathrm{View}_{V^*}[\langle P, V \rangle(x)]$ is what V^* sees when interacting with P.

$\mathrm{Sim}(x)$ is the algorithm that writes down the transcript without interacting with P.

Remember: input to $\mathrm{Sim}(\cdot)$ essentially captures what the (P, V) interaction leaks because that's the information the verifier is allowed to use when writing down the transcript.

How do we achieve ZK? What languages have ZK proofs?

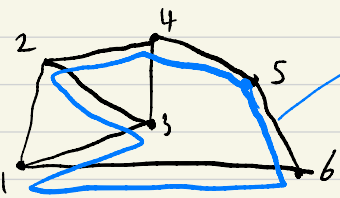
Will prove that there is a ZK proof protocol for every language in NP!

Approach: give a ZK protocol for one NP-complete problem (HAMCYCLE), then a ZK protocol for any other NP language is just to reduce the instance to this language and use the same protocol.

ZK protocol for Hamiltonian Cycle

Definition of hamiltonian cycle

Undirected graph $G = (V, E)$



Cycle in graph that visits each vertex once

HAMCYCLE = $\{G \mid G$ has a hamiltonian cycle$\}$

For our protocol, we will represent graphs as an adjacency matrix

$$G = \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 1 & 0 & 0 \\ 3 & 1 & 1 & 0 & 1 & 0 & 0 \\ 4 & 0 & 1 & 1 & 0 & 1 & 0 \\ 5 & 0 & 0 & 0 & 1 & 0 & 1 \\ 6 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

Matrix G s.t.

$$G_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in E(G) \\ 0 & \text{otherwise} \end{cases}$$

Trivial protocol

$P(G)$ $\qquad\qquad\qquad\qquad$ $V(G)$

$\xrightarrow{\text{edges on cycle}}$

Check that edges form a hamiltonian cycle

$\downarrow$

accept/reject

This protocol is complete and sound, but Probably not ZK.

Technicality: this actually might be ZK.

If $P = NP$, then V can compute the edges on the cycle by itself, so this would be ZK!

ZK protocol

Prover(G) | Verifier(G)

Prepare a bunch of commitments:

- $C_1, \dots, C_n$ commit to vertices $v_1, \dots, v_n$ in random order
- C_{ij} for $i,j \in [n]$ s.t. C_{ij} commits to $\begin{cases} 1 & \text{if vertices in } C_i \text{ and } C_j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$

C_i's = relabeling of vertices

C_{ij}'s = adjacency matrix under relabeling

$\xrightarrow{\text{Send the } n+n^2 \text{ commitments}}$

flip a coin $b \xleftarrow{R} \{0,1\}$

$\xleftarrow{\text{if } b=0 \text{ "show me } G\text{!" if } b=1 \text{ "show me the cycle!"}}$

if $b=0$, open all $n+n^2$ commitments

if $b=1$, open only commitments to C_{ij} in hamiltonian cycle in G

$\xrightarrow{\quad}$

if $b=0$
check if got a permutation of G's adjacency matrix

if $b=1$
check if got a cycle of length n

$\downarrow$

accept/reject

Assuming perfectly binding & computationally hiding commitment scheme.

Proof

1. Complete ✓

2. Sound — if $G \notin$ HAMCYCLE, then no matter what P^* commits to, V will reject w/prob $1/2$

   Otherwise it either has a hamiltonian cycle in G, or it can break the binding of the commitment. either of those would be a contradiction.

3. Zero Knowledge — Construct eff. Simulator

   $Sim(G)$
   - Guess $\hat{b} \xleftarrow{R} \{0,1\}$
   - if $\hat{b}=0$, commit to random permutation of G
     $\hat{b}=1$, commit to a random permutation of a length n cycle
   - Run $b \leftarrow V^*(G, \text{commitments})$
   - if $b \neq \hat{b}$, go back to start
   - else, open boxes per V^*'s request
   - Output transcript $(G, \text{commitments}, b, \text{commitment openings})$

Idea is to guess the bit b the verifier will ask for and prepare a response that only works for that choice. If it fails, retry. The final transcript looks the same as real protocol execution.

But we're not quite done!

Need to show $\{View_{V^*}[\langle P, V^* \rangle(G)]\} \approx_c \{Sim(G)\}$

How to do this? Use a hybrid argument!

Hybrid argument

Hybrid 0: $\{ View_{V^*}[\langle P, V^* \rangle(G)] \}$

$$\{ (G, C_1, \ldots, C_{n+n^2}, b, \text{commitment openings} \}$$

- identically distributed b/w real protocol & Sim
- in real protocol, this is a commitment to a shuffled graph G, but in Sim, this is sometimes a commitment to just a cycle

Hybrid i: replace the ith commitment with the corresponding commitment in the output of $Sim(G)$

$$\{ (G, C_1, C_2, \ldots, C_{i-1}, C_i, C_{i+1}, \ldots, C_{n+n^2}, b, \text{commitment openings} \}$$

Each hybrid can be proven computationally indistinguishable from the preceding hybrid by the hiding property of the commitment scheme.

proof idea: use adversary that distinguishes b/w two subsequent distributions to break hiding property of commitment.

Hybrid $n+n^2$: $\{ Sim(G) \}$

Done!!