Lecture 4: Zero Knowledge

Henry Corrigan-Gibbs
CS355 - Spring 2019
Plan

- Recap: Interactive proofs
- Zero Knowledge
  * What it is
  * Why it's useful
  * How we define it
- Example: ZK Proof for HAMCYCLE

Reminders

→ HW 1 due Friday at 5pm via Gradescope
→ Come to OH today?
→ Late day policy

Today

- We will be discussing the most beautiful idea in all of CS. Maybe of all time?
  * Controversial but still true:
- Zero knowledge - How to prove to you that I know something (e.g. $d$ is SAT) without leaking anything else to you (SAT assignment)
- Amazingly clever, also useful in many crypto protocols.
- Lesson: Importance of definitions.
  Original ZK paper is important b/c of defn of ZK, not because of the specific constructions.
  → Defn is >½ the battle
- Paper rejected 3 (I think) times before published
  → Lesson?
  Goldwasser, Micali, Rackoff (STOC '85)
Recap: Interactive proofs

On Monday, Florian introduced interactive proofs.

Goal of a proof: Convince someone of something—the verifier: statement

In complexity theory, we consider statements of the form:

"\( x \in L \)"

instance language

Examples: "\( N \) is the product of exactly two primes"

\( N \in \{ p q \mid \text{primes } p, q \} \)

"The Pythagorean Thm is true."

\( \text{PYTHM} \in \{ \text{true statements in some formal system} \} \)

"\( \phi \) is an unsatisfiable SAT formula"

\( \phi \in \{ \text{set of unsatisfiable SAT instances} \} \)
Recap

Conventional Proof

- It might be hard to find (exponential time)
- It should be easy to check (polynomial time, deterministic)

Say that we want to prove that $x \in L$.

Can do so w/ a conventional proof $\iff L$ is on NP language

E.g. to prove that $\phi$ is SAT, $P$ sends satisfying assignment to $V$.

Blum $NP = \text{"nifty proof"}$
Recap
What if we allow \( P \) \& \( V \) to interact?
What if \( V \) can use randomness?

**Prover \( (x) \)**

**Verifier \( (x) \)**

- Can increase to 1.
- "accept" or "reject"

**Properties we want**

1. Completeness
   \[ \forall x \in L \quad \Pr[\langle P, V \rangle(x) = "accept"] \geq \frac{2}{3} \]

2. Soundness
   \[ \forall x \notin L \quad \forall \bar{P} \quad \Pr[\langle \bar{P}, V \rangle(x) = "accept"] \leq \frac{1}{3} \]

Can reduce to negligible w/ repetition.
Q: Why is interaction useful?

A1: (On Monday) IP captures a larger class of problems. → PSPACE. . . prove to you that a graph is not 3-COLORABLE!

A2: (Today) Interactive proofs can have a third surprising property.

Properties we want

3. Zero knowledge V “learns nothing” from her interaction with P, except that x ∈ L.

Huh? What does this even mean?

Application: Can prove to you that I executed some protocol correctly without revealing any of my secrets.

Defn of ZK used to define security in many protocols want to show that “nothing leaks”
Q: What does it mean to “learn nothing” from an interaction?

Ex. Me in 7th grade

Me  "How was school today?"
   "Fine"

Dad

Ex. Military spokesperson.

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

Q: “How was school today?”
A: “Fine.”

If you can simulate the core transcript, no need to have it at all! Applications in real life?
The surprising thing is that there is a very clean way to formalize this intuition.

3. Zero knowledge: \( \forall \text{ efficient } V^* \exists \text{ efficient } \text{Sim} \)
\[
\left\{ \text{View}_{V^*}(x) \rightarrow V^*(x) \right\} \equiv \left\{ \text{Sim}(x) \right\}
\]

Intuition:
- Whatever \( V \) can learn by interacting with \( P \), it can learn sitting at home by running \( \text{Sim} \).
- Holds even if \( V^* \) is malicious.
- Key to remember: Input to \( \text{Sim} \) essentially captures what the \((P,V)\) interaction leaks.

There is an annoying technical issue that comes up when you want to run a \( ZK \) protocol many times.

\( \rightarrow \) "Auxiliary-input \( ZK \)"

See Goldreich §4.3.3.
**ZK Protocol for Hamiltonian Cycle** [Blum '87 (?)]

- HamCycle is an NP-Complete problem
- Anything provable (in NP) is provable in ZK
- Reduce to HamCycle instance, use this protocol.

In theory, can prove to you that I know an Oday in iOS without revealing it to you? And so on...

**Reminder:** Definition of Ham Cycle

\[ G = (V, E) \] undirected graph

Cycle in graph that visits each vertex once

See Knuth (linked from course website) for fun history of this problem.

\[ \text{HamCycle} = \{ G \mid G \text{ has a Hamiltonian cycle} \} \]

**Adjacency Matrix**

\[
A = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & & 1 & 1 & 0 & 0 \\
2 & & 1 & 0 & 1 & 1 \\
3 & & & 1 & 0 & 0 & 0 \\
4 & & & & 1 & 1 & 0 \\
5 & & & & & 1 & 0 \\
6 & & & & & & 1
\end{bmatrix}
\]

\[
A_{ij} = \begin{cases} 
1 & \text{if } (i, j) \in E(G) \\
0 & \text{otherwise}
\end{cases}
\]
Trivial Protocol

\[ P(G) \quad \text{edges on cycle} \quad V(G) \quad \text{check that edges make Ham cycle} \quad \text{acc/rej} \]

Not

Zero knowledge
(under reasonable assumptions...)
**ZK Protocol (Blum)**

Following Blum, we'll imagine that $P$ can send $V$ "locked boxes," which we implement w/ cryptographic commitments.

**Prover ($G$)**
* Put each of the $n$ vertices $v_1, ..., v_n$ into $n$ boxes $B_1, ..., B_n$ in random order.

**Verifier ($G$)**

* Into box $B_{ij}$, put $1$ if vertices in $B_i$ and $B_j$ are adjacent in $G$.

\[ B_i : \text{r labeling of vertices} \]
\[ B_{ij} : \text{adj. matrix under relabeling} \]

Send the $n + \binom{n}{2}$ boxes

Flip a coin $b \in \{0, 1\}$

If $b = 0$: "Show me $G$!"

If $b = 1$: "Show me the cycle."

If $b = 0$: Unlock all boxes.

If $b = 1$: Unlock only boxes corresponding to Ham Cycle in $G$.

**Check:**
* $b = 0$: Got a perm of adj. matrix keys
* $b = 1$: Got a cycle

Accept if so.
Some Comments

Imagine: \[ \text{Box} \] contains \( \text{msg} \) \( \rightarrow \) \( H(m, r) \)

\[ \text{Key} \] \( \rightarrow \) \( (m, r) \)

Properties

1. Complete. \( \checkmark \)

2. Sound.
   - If \( G \in \text{Ham Cycle} \), then no matter what \( P \) puts in boxes, \( V \) will reject w.p. \( \geq \frac{1}{3} \).

3. Zero knowledge. We construct eff. Sim.

\( \text{Sim}(G \in \text{Ham Cycle}) \)

- Guess \( b \leftarrow \{0, 1\} \).
- If \( b = 0 \), put random perm of Adj not in boxes
- If \( b = 1 \), put random perm of cycle in boxes.
- Run \( b \leftarrow V^*(G, \text{Boxes}) \)
- If \( b \neq b^* \), Abort.
- Else, open boxes per \( V^* \)'s request
- Output \( (G, \text{Boxes}, b, \text{Keys to boxes}) \) as transcript.

\[ \text{N.B.} \text{ When we replace ideal box with a real commitment, we get a protocol that is only computational } \text{ZK}. \]
Life lessons to remember

* If you can simulate an interaction, you haven't learned anything useful from it.
  (Ideally doesn't apply to this lecture.)

* Input to simulator ≤ what leaks.

* Anything that has a traditional (NP) proof also has a zero-knowledge proof system.