LECTURE 10

LAST TIME

Interactive proofs for statements $x \in L$

Verifier — Computationally bounded (poly time)
Can flip (private) random coins

Prover — All-powerful but capricious

Want protocols such that

COMPLETENESS $x \in L \Rightarrow \exists$ Prover that convinces verifier with high probability

SOUNDNESS $x \notin L \Rightarrow \forall$ Provers probability that verifier is fooled is low

Interactive proofs with public coins
Arthur-Merlin

All languages provable in constant # rounds in fact have 2-round Arthur-Merlin protocol:

1. Arthur flips coins and sends random string
2. Merlin gives answer
3. Arthur decides whether $x \in L$

IP = interactive proofs with # rounds polynomial in size of input

IP = PSPACE (!)

Proved weaker result $coNP \subseteq IP$

Key idea: Arithmetization

CNF formula $\phi \rightsquigarrow$ polynomial P_ϕ

Evaluate polynomial "step by step"
in large field $\Rightarrow$ makes it practically impossible for prover to cheat.

Can also define MULTIPROVER INTERACTIVE PROTOCOLS (MIP)

Provers agree beforehand on shared strategy
Cannot communicate during protocol

Can allow polynomially many provers
(verifier has to have time to read all answers)

In fact, just going from 1 to 2 provers gives as much power as poly many provers
Define MIP analogously to IP

Clearly $IP \subseteq MIP$ [can always ignore one prover]

Thm 1 [Babai, Fortnow, Lund '90]
$MIP = NEXP$

Why 2 provers more useful? Can use 2nd prover to force nonadaptivity of 1st prover

Suppose we ask prover 1 questions

$q_1, q_2, \ldots, q_m$

Prover 1 sees context and can choose answers to q_j accordingly.

But suppose we randomly pick $i \in [m]$ and ask Prover 2 q_i, and then require that P1 & P2 answers consistently on q_i

Then P1 can no longer answer adaptively

Might as well write down (big table with answer to all possible questions and publish

Verifier questions: random look-ups in table

$PCP[r, q]$ = set of languages that can be decided by q random checks in table of size 2^r [informal def]

Can restate Thm 1 as

$$NEXP = PCP[poly, poly] = \bigcup_{c \in \mathbb{N}^+} PCP[n^c, n^c]$$

Can be "scaled down" to

$$NP = PCP[polylog, polylog]$$

And further improved (with lots of work) to

THM 2 PCP THEOREM [Arora-Safra '92]
[Arora-Lund-Motwani-Sudan-Szegedy '99]

$$NP = PCP[O(\log n), O(1)]$$

Means that for any language L in NP
Can write down proofs π of $x \in L$ such that

- π has polynomial size in input x
- π can be checked by reading constant # bits of π independent of size of x!
- If $x \in L$ accept whp
  If $x \notin L$ reject whp

Now if this isn't magic...
More about this in guest lectures
Oct 8 and Oct 9 by PER AUSTRIN

Example 3 GRAPH NON ISOMORPHISM $\in$ PCP(poly(n), O(1))

$$GNI = \{ \langle G_0, G_1 \rangle \mid G_0 \not\cong G_1 \}$$

Graphs on n vertices

Represent graph by adjacency matrix

(Binary) string of length $n^2 \longleftrightarrow$ number in $[0, 2^{n^2}-1]$

Proof Binary string of length 2^{n^2}

Let position p correspond to graph H_p

Bit in position p is

a) 0 if $H_p \cong G_0$

b) 1 if $H_p \cong G_1$

c) arbitrary otherwise

Verifier test

1. Flip $b_R \in \{0, 1\}$
2. Construct random permutation $\pi : [n] \to [n]$
3. Let $H_p = \pi(G_b)$
4. Look up bit b' in position p
5. Accept if $b = b'$, reject otherwise

Analysis

Completeness If $G_0 \not\cong G_1$, Paver constructs legal proof table. Verifier's test will always accept.

Soundness If $G_0 \cong G_1$, then probability of checking position p independent of b

So, mentally, can construct π; $H_p = \pi(G_0)$, say; error check b' and only then flip coin $\Rightarrow$ exactly 1/2 soundness

(REST OF) TODAY : Cryptography

Just scratch the surface

DD2448 Foundations of Cryptography given in spring

"Human ingenuity cannot concoct a cipher which human ingenuity cannot resolve."
Edgar Allan Poe 1841

Cat-and-mouse game throughout the ages

Shannon (late '40s): rigorous definition of security

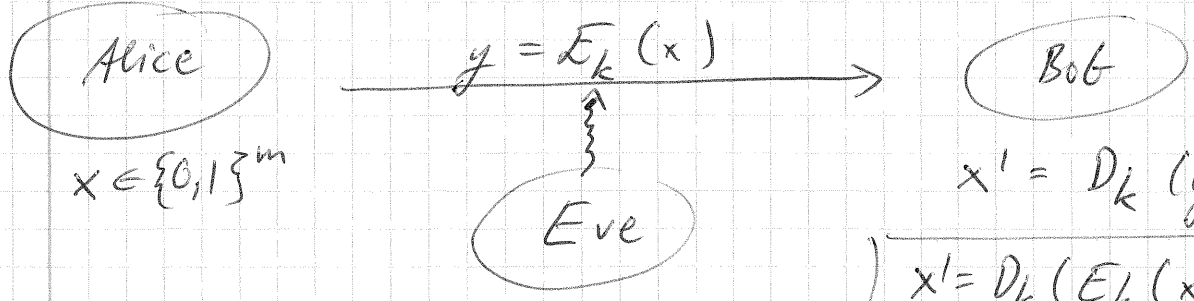
1970s: Birth of modern cryptography

Connection to computational complexity theory: Make code breaking a computationally hard problem (so hardness ⇒ good news!)

Cross-fertilization: Many ideas from crypto have turned out to be extremely useful in complexity theory

Basic task

Key $k \in_R \{0,1\}^n$



$$x' = D_k(E_k(x)) = x$$

DEF Encryption scheme (E, D) is PERFECTLY SECRET if $\forall x, x' \in \{0,1\}^m$, distributions $E_{U_n}(x)$ and $E_{U_n}(x')$ are identical

U_n = uniformly random n-bit strings

Ex 5 Suppose $n = m$

Let y = bitwise XOR of x and k

$(x_i + k_i \mod 2)$

One-time pad

Not hard to prove $E_k(x)$ looks perfectly random to outside observer

Claim 6 If (E, D) is an encryption scheme with $n < m$, then it's not perfectly secret

Proof Nice exercise.

Solution ?! Drop perfect secrecy.

Require secrecy only w.r.t computationally bounded adversaries

[Even NSA is computationally bounded...]

If this is to be possible, need $P \neq NP$

[See Lem 9.2 in Arora-Barak]

But this is not sufficient

Let us very briefly sketch a basic assumption of modern crypto and some consequences of it that allow us to recover a "computationally secure one-time pad"

DEF 7 Function $\varepsilon : \mathbb{N} \to [0,1]$ is NEGLIGIBLE if $\varepsilon(n) = n^{-\omega(1)}$

[That is, for every c there is an N s.t. for every $n > N$ $\varepsilon(n) < n^{-c}$]

DEF 8 A poly-time computable function $f: \{0,1\}^* \to \{0,1\}^*$ is a ONE-WAY FUNCTION if for every probabilistic poly time algorithm A there is a negligible function ε s.t.

$$\Pr_{\substack{x \in_R \{0,1\}^n \\ y = f(x)}} \left[A(y) = x' \text{ s.t. } f(x') = y \right] < \varepsilon(n)$$

CONJECTURE / ASSUMPTION 9 One-way functions exist

CLAIM 10 If one-way functions exist, then $P \neq NP$

Proof Good exercise; not hard.

CANDIDATE 11 Treat x as two $n/2$-bit integers A, B

Let $f(x) = A \cdot B$ = n bit number

Factoring number N can be done in time $\sqrt{N}$ but N is exponential in size of representation ($\log N$ bits).

Can set this problem up more carefully using primes P, Q

RSA function (as in the crypto system) another candidate.

INFORMAL THEOREM 12

If one-way functions exist, then computationally secure encryption schemes exist.

Proof idea Given random string $k \in \{0,1\}^n$
"Stretch it out" to pseudorandom string in $\{0,1\}^m$
Do one-time pad

What does "pseudorandom" mean?

(1) KOLMOGOROV COMPLEXITY
"A string of length n is random if no TM whose description is $< 0.99n$ size outputs this string when started on a blank tape"
"Right" definition, but (almost) totally useless

(2) STATISTICS
A string is random if it has the "right numbers" of substrings of different types (00, 11, 01, 10, 000, etc) as compared to what a random string has by laws of statistics
Too weak definition!

(3) CRYPTOGRAPHY
Focus on distributions over strings
A distribution is pseudorandom if it is indistinguishable from a truly uniformly random distribution to every probabilistic polynomial-time algorithm

DEF 13 A poly-time computable function $G : \{0,1\}^* \to \{0,1\}^*$ mapping n-bits strings to $\ell(n)$-bit strings, $\ell(n) > n$, is a SECURE PSEUDORANDOM GENERATOR OF STRETCH $\ell(n)$ if $\forall$ probabilistic poly-time A $\exists$ negligible $\varepsilon(n)$ such that $\forall n \in \mathbb{N}^+$

$$\left| Pr\left[A(G(U_n)) = 1 \right] - Pr\left[A(U_{\ell(n)}) = 1 \right] \right| < \varepsilon(n)$$

A is being fed pseudorandom bits

A is being fed truly random bits

can't tell the difference

THM 14 [Håstad, Impagliazzo, Luby, Levin '99]

If one-way functions exist, then secure pseudorandom generators exist for any polynomial stretch.

Lots of nice math in here...

Read Secs 9.2 - 9.3 in Arora-Barak if you are interested.

We won't cover any of this in the course (except the surface scratching in today's lecture).

Instead, let's switch topics back to interactive proofs.

Zero Knowledge Proofs

Last lecture: discussed interactive proofs
"We" were the verifier
Wanted to benefit from, but not be fooled by, computationally powerful prover.

Today: Switch the tables
Suppose "we" are the prover
Want to convince verifier we can prove $x \in L$, but verifier should learn no more than that

Ex Prove that we know a secret password without revealing anything about that password

Sounds a bit like science fiction... but, amazingly, this can be done! [Avra-Barak style]

DEF 15 Zero-knowledge proof
$L \in NP$, M verifier of certificates (poly-size) for $x \in L$
A pair P, V of interactive probabilistic poly-time algorithms is a zero knowledge proof for L if following 3 conditions hold:

COMPLETENESS $\forall x \in L$ and u s.t. $M(x,u) = 1$
$$Pr\left[out_V \langle P(x,u), V(x) \rangle = 1\right] \geq 2/3$$

SOUNDNESS If $x \notin L$ then for (every (all-powerful)) P^* any u and
$$Pr\left[out_V \langle P^*(x,u), V(x) \rangle = 1 \right] \leq 1/3$$

(PERFECT) ZERO KNOWLEDGE For every interactive probabilistic poly-time V^*, there exists an algorithm S^* which is probabilistic and runs in expected poly time such that $\forall x \in L$ $\forall u$ s.t. $M(x,u) = 1$

$$\text{out}_{V^*}\langle P(x,u), V^*(x)\rangle \equiv S^*(x) \qquad (+)$$

That is, the two random variables in (+) are identically distributed although S^* doesn't get access to u and doesn't even interact with P.

Formalizes meaning of "learning nothing"
Since we can reproduce transcripts without even talking to prover, conversation can't be leaking info.
This holds even for cheating verifier V^* who doesn't follow protocol.

Flavours of zero knowledge

(1) Perfect Simulator S^* gets transcript distribution exactly right

(2) Statistical: Transcript distributions (extremely) close in statistical distance

(3) Computational Transcript distributions are computationally indistinguishable

- Guarantees stronger in (1) than in (2) than in (3)
- Potentially more languages have zero-knowledge (ZK) protocols as in (3) than in (2) than in (1)

THM 16 If one-way functions exist, then *every* $L \in NP$ has a computational ZK proof.

Simulation Central concept in cryptography

Important in security definitions

Attacker cannot learn or do anything that isn't also possible in "sandbox" (which is obviously secure)

EX 17 GRAPH ISOMORPHISM $= GI = \{ (G_0, G_1) \mid G_0 \cong G_1 \}$

Public input G_0, G_1 ~~graphs~~ on n vertices

Prover's private input: Permutation $\pi : [n] \to [n]$ s.t. $G_1 = \pi(G_0)$

Protocol

1. Prover: Choose random permutation $\pi_2 : [n] \to [n]$
   Send $H = \pi_1(G_2)$ to verifier
2. Verifier: Choose $b \in_R \{0,1\}$; send to prover
3. Prover: If $b=1$, send permutation π_1
   If $b=0$, send permutation $\pi_1 \circ \pi$
   Let sent permutation be π_H
4. Verifier: Check that $H = \pi_H(G_b)$
   Accept if yes, Reject if no.

Analysis

Completeness If $G_0 \cong G_1$ and P & V follow protocol, V will accept with probability 1.

Soundness $G_0 \not\cong G_1$. Then $\exists b \in \{0,1\}$ s.t. $H \not\cong G_b$

Verifier will choose this b with probability $1/2$

If so, Prover must fail check in step 4.

Zero knowledge Fix (misbehaving) verifier V^* and construct simulator S^* as follows

(i) Choose $b' \in_R \{0,1\}$ and random $\pi : [n] \to [n]$

Compute $H = \pi(G_{b'})$ and feed into V^*

(which we can run as subroutine)

(ii) Let $b \in \{0,1\}$ be message from V^*

(iii) If $b \neq b'$ then abort and restart from (i)

else

feed π into V^*

Let final transcript be

"H, b, π, V^*'s final output"

Need to argue:

(a) S^* runs in expected poly time

(b) Transcript distribution is correct

(a) S^*'s first message has exactly right distribution

Reveals nothing about b' (information-theoretically)

Hence $Pr[b = b'] = 1/2$

Expected # attempts before S^* succeeds

$\sum_{i=1}^{\infty} i \cdot Pr[i \text{ attempts needed}] =$

$\sum_{i=1}^{\infty} \frac{i}{2^i} = O(1)$

(b) Suppose S^* terminates, then:

H is random permutation of G_1

b is a random bit

π is correct (uniquely determined by H and b)

And hence V^* final message also correct.

~~~

This concludes first half (roughly) of course

Most of what you "have to know" about computational complexity theory so that we can credibly claim we give you a well-rounded education

[Major omission: Quantum computation]

Remaining lectures: selection (very biased) of some "advanced topics" [but not necessarily technically harder]

- proof complexity
- property testing
- PCPs and hardness of approximation
- Communication complexity

} all research areas in their own right
~~~