Interactive proofs
Adding interactivity + randomization to polynomial time \( \Rightarrow \text{PSPACE} \)

Probabilistically checkable proofs (PCPs)
\[ \text{NP} = \text{PCP}[O(\log n), O(1)] \]

NP has polynomial size proofs that can be checked reading constant # bits (and with small risk of error)

Cryptography

key \( k \in \{0,1\}^m \)

\( x \in \{0,1\}^n \)

Alice

\[ y = E_k(x) \]

Bob

\[ x' = D_k(y) \]

\[ x' = D_k(E_k(x)) = x \]

Eve

Notion of secrecy: For two messages \( x_1, x_2 \), distributions \( E_k(x_1) \) and \( E_k(x_2) \) should "look the same" to Eve

For \( n = m \): ONE-TIME PAD \( y = x \oplus k \)
Perfect secrecy

For \( n < m \), perfect, information theoretic secrecy is impossible
Solution: Aim for secrecy against computationally bounded adversaries.

**Def 17** \( \varepsilon : \mathbb{N} \rightarrow [0, 1] \) is **negligible** if
\[
\varepsilon(n) = n^{-o(1)} \quad [ \text{decays faster than } 1/p(n) \text{ for any polynomial } p(n) ]
\]

**Def 18** \( f : \{0, 1\}^* \rightarrow \{0, 1\}^* \) is a **one-way function** if
a) \( f \) poly-time computable
b) \( f \) is hard to invert in the sense that a probabilistic poly-time \( A \) \( \exists \) negligible \( \varepsilon \) s.t.
\[
\Pr_{x \in \{0, 1\}^n, y \leftarrow f(x)} \left[ A(y) = x' \quad \text{s.t.} \quad f(x') = y \right] < \varepsilon(n).
\]

**Conjecture/Assumption 19** One-way functions exist.

**Claim 20** If one-way functions exist, then \( P \neq NP \).

**Candidate 21** Take \( x \in \{0, 1\}^n \) for \( n = 2n' \).
View \( x \) as concatenation of two \( n' \)-bit integers \( A \) and \( B \).
Let \( f(x) = A \cdot B = n \)-bit number.
Factoring number \( N = A \cdot B \) can be done in time \( \sqrt{N} \).
But this is exponential in size of input \((\log N \text{ bits})\).
Can set up this more carefully requiring $A = P$ and $B = Q$ to be prime numbers.

Another candidate is the function used in the RSA cryptosystem.

INFORMAL THEOREM 22

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

Proof idea: Given truly random string $k \in \{0,1\}^n$, "stretch it out" to pseudorandom string $k' \in \{0,1\}^{m'}$.

Then do one-time pad with this string $k'$.

What does "pseudorandom" mean?

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

2. STATISTICS
   
   A string is random if it has expected # substrings of different types (0, 1, 00, 01, 10, 11, 000, etc.) for truly random string. TOO WEAK definition!
Cryptography

Focus on distributions over strings
A distribution is pseudorandom if it is indistinguishable from truly uniformly random distribution for every probabilistic polynomial-time algorithm.

DEF 2.3 A poly-time computable function $G : \{0,1\}^* \rightarrow \{0,1\}^*$ mapping $n$-bit strings to $t(n)$-bit strings is a [SECURE PSEUDORANDOM GENERATOR] of stretch $t(n)$ if a probabilistic poly-time $A \exists \epsilon(n)$ s.t. $\forall (n) \epsilon N^+$ such that

$$\text{Pr}[A(G(U_n)) = 1] - \text{Pr}[A(U_{t(n)}) = 1] < \epsilon(n)$$

A gets pseudorandom bits, A gets truly random bits, can't tell difference.

THM 2.4 [H˚astad, 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 Sects 9.2 - 9.3 in Avra-Barak if you are interested (and then take our foundations of crypto course!)
We won't talk any more about this in this course (which is a computational complexity course after all...).

For the rest of this lecture, let's switch focus back to interactive proofs (but with a crypto perspective).

**Zero-Knowledge Proofs**

In interactive proofs so far, we've verified what we benefit from, but not been fooled by powerful but dangerous provers.

Now let's switch the tables.

Suppose we are the prover.

Want to convince verifier we can prove ACL (e.g., $F$ is satisfiable CNF).

But without leaking any more information (don't reveal anything about satisfying assignment to $F$).

Concrete example:

We want to authenticate by proving to verifier that we know secret password.

But don't want to reveal anything about that password.

Sounds a bit like science fiction...

But, amazingly, this can be done.
DEF 25  **ZERO-KNOWLEDGE PROOF**

Let \( L \subseteq \text{NP} \); \( M \) verifier of (poly-size) certificates for \( x \in L \)

A pair \((P, V)\) of interactive, probabilistic, poly-time algorithms form a **ZERO-KNOWLEDGE PROOF** for \( L \) if the following 3 conditions hold:

1. **Completeness**  \( \forall x \in L \) and \( u \) s.t. \( M(x, u) = 1 \)
   \[ \Pr\left[ \text{out}_V \left< P(x, u), V(x) \right> = 1 \right] \geq \frac{2}{3} \]

2. **Soundness**  \( \forall x \notin L \) \( \forall u \) \( \forall P^* \) (all-powerful)
   \[ \Pr\left[ \text{out}_V \left< P^*(x, u), V(x) \right> = 1 \right] \leq \frac{1}{3} \]

3. **(Perfect) Zero Knowledge**  \( \forall \) probabilistic, interactive poly-time \( P^* \) algorithm \( S^* \) such that
   - \( S^* \) runs in expected polynomial time
   - \( \forall x \in L \) \( \forall u \) s.t. \( M(x, u) = 1 \)
   \[ \text{out}_{V^*} \left< P(x, u), V^*(x) \right> \equiv S^*(x) \quad (\dagger) \]

That is, random variables in \((\dagger)\) identically distributed although \( S^* \) doesn't get access to \( u \) and doesn't even interact with \( P \)

Formalizes meaning of "learning nothing" since transcript can be reproduced without even talking to prover, conversation can't be leaking info
This should hold even for cheaply verifiable \( V^* \) who doesn't follow protocol [Verifier who follows protocol is "honest but curious"

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)

JHIM 26: If one-way functions exist, then every \( L \in \text{NP} \) has a computational ZK proof.

Simulatin: Central concept in cryptography

Important in security definitions: Attacker can't learn or do anything that isn't also possible in "sandbox" (which is obviously secure)
EXAMPLE 27  Just to see a ZK protocol, let's look at

\[
\text{GRAPH ISOMORPHISM (GI)} = \\
\{ (G_0, G_1) \mid G_0 \cong G_1 \}
\]

**Public input** \( G_0, G_1 \) graphs on \( n \) vertices

**Prover's private input** Permutative \( \pi : [n] \rightarrow [n] \)

\( \text{so } G_1 = \pi(G_0) \)

**Protocol**

1. **Prover:** Choose random permutation \( \pi : [n] \rightarrow [n] \)
   send \( H = \pi(G_1) \) to verifier

2. **Verifier:** Choose \( b \in \{0, 1\} \); send to prover

3. **Prover:** If \( b = 1 \), send \( \pi^{-1} \)
   If \( b = 0 \), send \( \pi^{-1} \circ \pi \)
   let sent permutation be \( \pi_H \)

4. **Verifier:** Accept if \( H = \pi_H(G_0) \);
   reject otherwise

**Analysis**

Completeness  If \( G_0 \cong G_1 \) as witnessed by \( \pi \), and if \( P \) and \( V \) follow protocol, then \( V \) accepts with probability 1.

Soundness  If \( G_0 \not\cong G_1 \), then \( \exists b \in \{0, 1\} \) s.t. \( H \neq G_b \)

Verifier chooses this \( b \) with probability \( 1/2 \)

in which case prove fails the test
in step 4 and verifier rejects
(Repeat protocol twice; refer to \( 1/4 < 1/3 \))

Zero knowledge. Consider a fixed (non-interactive) verifier \( V^* \) and construct simulator \( S^* \) as follows:

(i) Choose \( b' \in \{0,1\} \) and random \( \pi : [n] \rightarrow [n] \)
Compute \( H = \pi(6b') \) and feed into \( V^* \)

Important. Note that we can run \( V^* \) as a subroutine.
It is just a piece of code (and does not have a state or memory or anything)

(ii) Let \( b \in \{0,1\} \) be message from \( V^* \)
[and terminate protocol if \( V^* \) doesn't send a bit]

(iii) If \( b \neq b' \) then abort attempt at generating transcript and restart from (i)

Important. Note that this reboots \( V^* \) - since \( V^* \) is just a probabilistic TM it will happily run again without "knowing" that it has been restarted

else \( b = b' \)
Feed \( \pi \) into \( V^* \)
Let final transcript be
"\( H, b, \pi, V^* \)'s final output"
Need to argue

(a) Distribution of generated transcript is correct

(b) $S^*$ runs in expected polynomial time

(a) Suppose $S^*$ terminates. Then
- $H$ is random permutation of $G_1$
- $t_0$ is response from $V^*$ generated based on message with correct distribution
- $t_0$ is correct (as determined by $H$ and $G$)

Since distributions of all messages so far is the same as in real conversation between $V^*$ and $P$, the final message of $V^*$ will also have the correct distribution so the transcripts are identically distributed

(b) $V^*$ runs in poly time, so every attempt takes poly time for $S^*$

Expected # attempts before $S^*$ succeeds

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

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

So $S^*$ runs in expected polynomial time.

This concludes the analysis.
With this we wrap up the "compulsory" (roughly) first half of the course.

We have now covered most of what you "should know" about computational complexity theory after an introductory course.

This should hopefully mean that:

- We can credibly claim to have given you a reasonably well-rounded (computational complexity) education.
- You have a fighting chance of being able to read and understand papers that appear in, say, the Computational Complexity Conference (CCC) [which is useful if you want grade A].

But of course we haven't covered everything. Perhaps the one major omission is QUANTUM COMPUTATION.
2nd, "advanced part" of course
Selection (by necessity biased) of some more specialized topics
[ but not necessarily technically harder ]
Details decided as we go along
Probably:
- Communication complexity [already happened]
- Circuit complexity [ lots of news in last few years although we will look at classic results ]
- Proof complexity [ My area + nice connections to both circuit complexity and communication complexity ]
- And/or maybe some other stuff . . .