Focus thus far: protecting communication (e.g., message confidentiality and message integrity).

Next few weeks: protecting computations.

Zero-knowledge: a defining idea at the heart of theoretical cryptography
- Idea: will seem very counter-intuitive, but surprisingly powerful.
- Shows the importance and power of definitions (e.g., "What does it mean to know something?")

We begin by introducing the notion of a "proof system".
- Goal: A prover wants to convince a verifier that some statement is true
  e.g., "This Sudoku puzzle has a unique solution"
  "The number \( N \) is a product of two prime numbers \( p \) and \( q \)"
  "I know the discrete log of \( h \) base \( g \)

We model this as follows:

\[
\text{prover} (X) \xrightarrow{\pi} \text{verifier} (X) \quad X: \text{statement that the prover is trying to prove (known to both prover and verifier)}
\]

\[
\pi: \text{the proof of } X \quad \{ \text{there are all examples of statements} \}
\]

\[
b \in \{0,1\}^* \quad \{ \text{given statement } X \text{ and proof } \pi, \text{ verifier decides whether to accept or reject} \}
\]

Properties we care about:
- Completeness: Honest prover should be able to convince honest verifier of true statements
  \[ \forall X \in L: \Pr [\pi \leftarrow P(X) : V(X, \pi) = 1] = 1 \]
- Soundness: Dishonest prover cannot convince honest verifier of false statement
  \[ \forall X \notin L: \Pr [\pi \leftarrow P(X) : V(X, \pi) = 1] < \frac{2}{3} \]

Important: we are not restricting to efficient provers.

Typically, proofs are "one-shot" (i.e., single message from prover to verifier) and the verifier's decision algorithm is deterministic.
- Languages with these types of proof systems precisely coincide with \( \text{NP} \) (proof of statement \( X \) is to send \( \text{NP} \) witness \( w \)).

Going beyond \( \text{NP} \): we augment the model as follows
- Add randomness: the verifier can be a randomized algorithm.
- Add interaction: verifier can ask "questions" to the prover.

Interactive proof systems [Goldwasser-Micali- Rackoff]:

\[
\text{prover} (X) \xrightarrow{\pi} \xrightarrow{} \xrightarrow{} \xrightarrow{} \text{verifier} (X) \]

Set of languages that have an interactive proof system is denoted \( \text{IP} \).

Theorem (Shamir): \( \text{IP} = \text{PSPACE} \)

Languages that can be decided in polynomial space [very large class of languages!].
**Takeaway:** Interaction and randomness is very useful.

\[ \Rightarrow \quad \text{In fact, enables a new property called zero-knowledge} \]

Consider the following example: Suppose prover wants to convince verifier that \( N = p^2 \) where \( p,q \) are prime (and secret). We have:

\[ \text{prover \((N, p, q)\)} \quad \text{verifier \((N)\)} \]

\[ N = p^2 \quad \downarrow \]

accept if \( N = p^2 \) and reject otherwise.

Proof: certainly complete and sound, but now verifier also learned the factorization of \( N \). (May not be desirable if prover was trying to convince verifier that \( N \) is a proper RSA modulus (for a cryptographic scheme) without revealing factorization in the process.

\[ \Rightarrow \quad \text{In some sense, this proof conveys information to the verifier [i.e., verifier learns something it did not know before seeing the proof]} \]

**Zero-knowledge:** ensure that verifier does **not** learn anything (other than the fact that the statement is true).

**How do we define “zero-knowledge”?** We will introduce a notion of a “simulator.”

**Definition.** An interactive proof system \( \langle P, V \rangle \) is zero-knowledge if for all efficient (and possibly malicious) verifiers \( V^* \), there exists an efficient simulator \( S \) such that for all \( x \in L \):

\[ \text{View}^*_v(\langle P, V \rangle(x)) \approx S(x) \]

random variable denoting the set of messages sent and received by \( V^* \) when interacting with the prover \( P \) on input \( x \).

What does this definition mean?

\[ \text{View}_v(P \Rightarrow V^*(x)) : \text{this is what } V^* \text{ sees in the interactive proof protocol with } P \]

\[ S(x) : \text{this is a function that only depends on the statement } x, \text{ which } V^* \text{ already has} \]

If these two distributions are indistinguishable, then anything that \( V^* \) could have learned by talking to \( P \), it could have learned just by invoking the simulator itself, and the simulator output only depends on \( x \), which \( V^* \) already knows.

\[ \Rightarrow \quad \text{In other words, anything } V^* \text{ could have learned (i.e., computed) after interacting with } P, \text{ it could have learned without ever talking to } P! \]

**Very remarkable definition!**

**More remarkable:** If one-way functions exist, then every language \( L \in \text{IP} \) has a zero-knowledge proof system.

\[ \Rightarrow \quad \text{Namely, anything that can be proved can be proved in zero-knowledge!} \]

We will show this theorem for NP languages. Here it suffices to construct a single zero-knowledge proof system for an NP-complete language. We will consider the language of graph 3-colorability.

3-coloring: given a graph \( G \), can you color the vertices so that no adjacent nodes have the same color?
We will need a commitment scheme \((\text{Commit, Verify})\). A (non-interactive) commitment scheme consists of two main algorithms \((\text{Commit, Verify})\):

- \(\text{Commit}(m) \rightarrow (c, r)\): Takes a message \(m\) and outputs the commitment \(c\) and an opening \(r\).
- \(\text{Verify}(m, c, r) \rightarrow b\): Checks if \(c\) is a valid opening to \(m\) (with respect to opening \(r\)).

[The commitment scheme might also take public parameters \((\text{Commit, Verify})\), but for simplicity, we omit them / leave them implicit.]

\[\text{Completeness}: \] For all messages \(m\):
\[
\Pr[\text{Commit}(m) \rightarrow (c, r) : \text{Verify}(m, c, r) = 1] = 1
\]

\[\text{Soundness}: \] For all efficient adversaries \(A\), if \((m, m') \leftarrow A(\text{commit})\):
\[
\{ (c, r) \leftarrow \text{Commit}(m) : c \neq \text{Commit}(m', r) \}
\]

\[\text{Intuitively:}\] Prover commits to a coloring of the graph.

Verifier challenges power to reveal coloring of a single edge.

Power reveals the coloring on the chosen edge and opens the entries in the commitment.

[Completeness: By inspection, if coloring is valid, power can always answer the challenge correctly.]

[Soundness: Suppose \(G\) is not 3-colorable. Let \(K_1, \ldots, K_n\) be the coloring the power committed to. If the commitment scheme is perfectly binding, \(c_1, \ldots, c_n\) uniquely determine \(K_1, \ldots, K_n\). Since \(G\) is not 3-colorable, there is an edge \((i, j) \in E\) where \(K_i = K_j\) or \(i \notin \{0, 1, 2\}\) or \(j \notin \{0, 1, 2\}\). Otherwise, \(G\) is 3-colorable with coloring \(K_1, \ldots, K_n\). Since the verifier chooses an edge to check at random, the verifier will choose \((i, j)\) with probability \(1/|E|\). Thus, if \(G\) is not 3-colorable, \(\Pr[\text{verifier rejects}] > 1/|E|\).

Thus, this protocol provides soundness \(1 - 1/|E|\). We can repeat this protocol \(O(1/|E|^2)\) times sequentially to reduce soundness error to \(\Pr[\text{verifier accepts proof of false statement}] \leq (1 - 1/|E|)^2 \leq e^{-|E|} = e^{-m} \left(\text{since } (1 - 1/2)^2 \leq 1/2\right)\).]
Zero Knowledge: We need to construct a simulator that outputs a valid transcript given only the graph $G$ as input. Let $V^*$ be a (possibly malicious) verifier. Construct simulator $S$ as follows:

1. Choose $K_i \leftarrow \{0,1,2,3\}$ for all $i \in [n]$.
2. Let $(c_i, r_i) \leftarrow \text{Commit}(1^3, K_i)$.
3. Give $(c_1, \ldots, c_n)$ to $V^*$.

Simulator does not know coloring, so it commits to a random one.

1. $V^*$ outputs an edge $(i,j) \in E$.
2. If $K_i \neq K_j$, then $S$ outputs $(K_i, K_j, r_i, r_j)$.
3. Otherwise, restart and try again (it fails $\lambda$ times, then abort).

Simulator succeeds with probability $1/3$ (over choice of $K_1, \ldots, K_n$). Thus, simulator produces a valid transcript with prob. $1 - \frac{1}{3^\lambda} = 1 - \text{negl}(\lambda)$ after $\lambda$ attempts. It suffices to show that simulated transcript is indistinguishable from a real transcript:

- Real scheme: prover opens $K_i, K_j$ where $K_i, K_j \leftarrow \{0,1,2,3\}$ [since prover randomly permutes the colors]
- Simulation: $K_i$ and $K_j$ sampled uniformly from $\{0,1,2,3\}$ and conditioned on $K_i \neq K_j$, distributions are identical

In addition, $(i,j)$ output by $V^*$ in the simulation is distributed correctly since commitment scheme is computationally-hiding (e.g. $V^*$ behaves essentially the same given commitments to a random coloring as it does given commitment to a valid coloring).

If we repeat this protocol (for soundness amplification), simulator simulates one transcript at a time.