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

\[
\begin{array}{c}
\text{prover} (N,q) \\
\hline
\rightarrow \quad \pi = (p,q) \\
\downarrow \\
\text{accept if } N = pq \text{ and reject otherwise}
\end{array}
\]

Proof is 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.\

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

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.”

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

\[
\text{View}_{P} (\langle P, V \rangle (x)) \cong S(x)
\]

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?

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

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

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.

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

Very remarkable definition!\]

More remarkable: using cryptographic commitments, then every language \( L \in \text{IP} \) has a zero-knowledge proof system.

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

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.

\[ \text{3-colorable} \quad \text{and not 3-colorable} \]

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. A (non-interactive) commitment scheme consists of three algorithms (Setup, Commit, Open):

- **Setup** \( (1^k) \rightarrow \sigma \): Outputs a common reference string \( \sigma \) (used to generate/validate commitments).
- **Commit** \( (\sigma, m) \rightarrow (c, \pi) \): Takes the CRS \( \sigma \) and message \( m \) and outputs a commitment \( c \) and opening \( \pi \).
- **Verify** \( (\sigma, m, c, \pi) \rightarrow 0/1 \): Checks if \( c \) is a valid commitment to \( m \) (given \( \pi \)).

### Typical Setup:

<table>
<thead>
<tr>
<th>Committer</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sigma \leftarrow \text{Setup}(1^k) )</td>
<td>( \sigma )</td>
</tr>
<tr>
<td>( (c, \pi) \leftarrow \text{Commit}(\sigma, m) )</td>
<td>( c )</td>
</tr>
<tr>
<td>(sometime later)</td>
<td>( m, \pi )</td>
</tr>
<tr>
<td>can check that ( \text{Verify}(\sigma, m, c, \pi) = 1 )</td>
<td></td>
</tr>
</tbody>
</table>

### Requirements: [see HW5 for construction from OWFs]

- **Correctness**: for all messages \( m \):
  \[
  \Pr[\sigma \leftarrow \text{Setup}(1^k); (c, \pi) \leftarrow \text{Commit}(\sigma, m); \text{Verify}(\sigma, c, m, \pi) = 1] = 1
  \]

- **Hiding**: for all common reference strings \( \sigma \in \{0,1\}^k \) and all efficient A, following distributions are computationally indistinguishable:

  \[
  \begin{align*}
  & \text{adversary} & \text{challenger} \downarrow \\
  & \sigma & \rightarrow \text{commit}(\sigma, m, \pi) \\
  & m, m \downarrow & c \\
  & b' \in \{0,1\} \\
  \end{align*}
  \]

  \[
  \Pr[b' = 1 | b = 0] = \Pr[b' = 1 | b = 1] = \text{negl}(\lambda)
  \]

- **Binding**: for all adversaries A, if \( \sigma \leftarrow \text{Setup}(1^k) \), then:
  \[
  \Pr[(m, m, c, \pi, \pi_0) \leftarrow A : m \neq m \text{ and } \text{Verify}(\sigma, c, m, \pi_0) = 1; \text{Verify}(\sigma, c, m, \pi) = 1] = \text{negl}(\lambda)
  \]
A 2K protocol for graph 3-coloring:

- let \( K_i \in \{0,1,2\} \) be a 3-coloring of \( G \)
- choose random permutation \( \pi \) of \( \{0,1,2\} \)
- for \( i \in (n) \): 
  - \((c_i, \pi_i) \leftarrow \text{Commit}(\sigma, K_i)\)
  - for random \( r_i \):
  - \((K_i, r_i), (K_i, \pi_i) \) \leftarrow (c_i, \pi_i, r_i) \)
- reject if \((ij) \notin E\)

Verifier (\( G \))
- \( \sigma \leftarrow \text{Setup}(\tau^2) \)
- \((i,j) \in E \)
- accept if \( K_i \neq K_j \) and \( K_i, K_j \in \{0,1,2\} \)
- \( \exists \) \( (c_i, c_j, K_i, r_i) \in \text{Commit}(\sigma, K_i, K_j) \)
- \( \text{Verify}(\sigma, c_i, K_i, r_i) = 1 = \text{Verify}(\sigma, c_j, K_j, r_j) \)
- reject otherwise

Intuitively: Prover commits to a coloring of the graph.
Verifier challenges prover to reveal coloring of a single edge.
Prover reveals the coloring on the chosen edge and opens the entries in the commitment.

Completeness: By inspection, if coloring is valid, prover can always answer the challenge correctly except with prob. \( 1 - \frac{1}{\tau} \).

Soundness: Suppose \( G \) is not 3-colorable. Let \( K_1, \ldots, K_n \) be the coloring the prover committed to. If the commitment scheme is statistically 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 \((ij)\) with probability \( \frac{1}{|E|} \). Thus, if \( G \) is not 3-colorable,
\[
\Pr[\text{verifier rejects}] > \frac{1}{|E|}
\]
Thus, this protocol provides soundness \( 1 - \frac{1}{|E|} \). We can repeat this protocol \( O(|E|^2) \) times sequentially to reduce soundness error to
\[
\Pr[\text{verifier accepts proof of false statement}] \leq \left(1 - \frac{1}{|E|}\right)^{|E|^2} \leq e^{-\frac{1}{|E|}} = e^{-e} \leq e^{-e} \quad \text{[since } 1+x \leq e^x \text{]}
\]
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. Run $V^*$ to get $c^*$.
2. Choose $K_i \leftarrow \{0,1,2,5\}$ for all $i \in [n]$. Let $(c_1,\ldots,c_n)\leftarrow \text{Commit}(c^*,K_i)$.
   - Simulator does not know coloring: so it commits to a random one.
   - Give $(c_1,\ldots,c_n)$ to $V^*$.
3. $V^*$ outputs an edge $(i,j) \in E$.
4. If $K_i \neq K_j$, then $S$ outputs $(K_i,K_j,i,j)$.
   - Otherwise, restart and try again (it fails $\lambda$ times, then abort).

Simulator succeeds with probability $\frac{2}{3}$ (over choice of $K_{i_1},\ldots,K_n$). Thus, simulator produces a valid transcript with prob. $1 - \frac{1}{3^\lambda} = 1-\text{neg}(\lambda)$ 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 $K_i,K_j \leftarrow \{0,1,2,5\}$ [since prover randomly permutes the colors]
- Simulation: $K_i$ and $K_j$ sampled uniformly from $\{0,1,2,5\}$ 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 simulate one transcript at a time.

Summary: Every language in $\text{NP}$ has a zero-knowledge proof (assuming existence of OWFs).

Can be used to obtain ZK proof for $\text{IP}$:

(Without loss of generality, suppose proof is public-coin - e.g., an Arthur-Merlin proof)\[ \text{recall: IP}(k) \in \text{AM}(k+2) \]

To construct ZK proof for $L \in \text{IP}$, proceed as follows:

1. Replace prover’s message with a computationally-hiding and statistical binding commitment to message.
2. Verifier just send its random coins as in the AM protocol.
3. Prover proves in zero-knowledge at the very end that the set of messages it committed to would cause the verifier to accept.

This is an NP statement [witness is the commitment opening and messages, relation checks openings to commitment and that verifier accepts the transcript].

Implication: Everything that can be proven (IP) can be proven in zero knowledge!