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

- Hiding: for all common reference strings \( \sigma \in \{0,1\}^n \) and all efficient \( A \), following distributions are computationally indistinguishable:
  \[
  \Pr[b' = 1 | b = 0] - \Pr[b' = 1 | b = 1] = \text{negl}(\lambda)
  \]

- Binding: for all adversaries \( A \), if \( \sigma \leftarrow \text{Setup} \), then
  \[
  \Pr[(m, m, c, \pi, x) \leftarrow A : m \neq m \text{ and} \text{Verify}(\sigma, c, m, x) = 1] = \text{negl}(\lambda)
  \]

A 2K protocol for graph 3-coloring:

- let \( K \in \{0,1,2\} \) be a 3-coloring of \( G \)
- choose random permutation \( T \in \text{Perm}(\{0,1,2\}) \)
- for \( i \in \{n\} \):
  \[
  (c_i, x_i) \leftarrow \text{Commit}(\sigma, T(i))
  \]
  for random \( r \)
- reject if \((i,j) \notin E\)

\[
\begin{align*}
\text{prover}(G) & \quad \text{contains } n \text{ nodes, } \quad m \text{ edges} \\
\text{verifier}(G) & \quad \text{contains } n \text{ nodes, } \quad m \text{ edges} \\
\end{align*}
\]
Intrinsically: Prover commits to a coloring of the graph
Verifier challenge provers to reveal coloring of a single edge
Verifier 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 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 \(i \not\in \{0,1,2\}\) or \(j \not\in \{0,1,2\}\). [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,

\[
\Pr[\text{verifier rejects}] \geq \frac{1}{|E|}
\]

Thus, this protocol provides soundness \(1 - \frac{1}{|E|}\). We can repeat this protocol \(O(|E|^2)\) times sequentially to reduce soundness error to

\[
\Pr[\text{verifier accepts proof of false statement}] \leq (1 - \frac{1}{|E|})^{1/|E|} \leq e^{-|E|/|E|} = e^{-1} = e^{-m}\quad \text{[since } 1 + x \leq e^x]\]

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 \(\sigma^*\).
2. Choose \(K_i \in \{0,1,2\}\) for all \(i \in [n]\).
   
   \(\{\sigma_i^{(i)} \rightarrow \text{Commit}(\sigma^*, K_i)\}\)  \forall i \in [n]
   
   Give \((\sigma_1, \ldots, \sigma_n)\) to \(V^*\).
3. \(V^*\) outputs an edge \((i,j) \in E\).
4. If \(K_i \neq K_j\), then \(S\) outputs \((K_i, K_j, \pi_i, \pi_j)\).

Otherwise, restart and try again (if fails \(\lambda\) times, then abort).

Simulator succeeds with probability \(\frac{3}{5}\) (over choice of \(K_1, \ldots, K_n\)). Thus, simulator produces a valid transcript with prob. \(1 - \frac{1}{5^2} = 1 - \text{negl}(n)\) 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 \(K_i, K_j \in \{0,1,2\}\) [since power randomly permutes the colors].
- Simulation: \(K_i, K_j\) sampled uniformly from \(\{0,1,2\}\) and conditioned on \(K_i, K_j\) 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 \(NP\) has a zero-knowledge proof (assuming existence of PRGs)

\(\uparrow\)

PRGs imply commitments
In many cases, we want a stronger property: the prover actually "knows" why a statement is true (e.g., it knows a "witness").

For instance, consider the following language:

\[ L = \{ h \in G \mid \exists x \in \mathbb{Z}_p^*: h = g^x \} = G \]

Note: this definition of \( L \) implicitly defines an NP relation \( R \):

\[ R((h, x) = 1) \iff h = g^x \in G \]

In this case, all statements in \( G \) are true (i.e., contained in \( L \)), but we can still consider a notion of proving knowledge of the discrete log of an element \( h \in G \) — conceptually stronger property than proof of membership.

Philosophical question: What does it mean to "know" something?

If a prover is able to convince an honest verifier that it "knows" something, then it should be possible to extract that quantity from the prover.

**Definition.** An interactive proof system \((P, V)\) is a proof of knowledge for an NP relation \( R \) if there exists an efficient extractor \( E \) such that for any \( x \) and any prover \( P^* \)

\[ Pr[\omega \leftarrow E^*(x) : R(x, \omega) = 1] \geq Pr[<P^*, V>(x) = 1] - \varepsilon \]

more generally, could be polynomially smaller

\[ \text{knowledge error} \]

**Trivial proof of knowledge:** prover sends witness in the clear to the verifier

\[ \rightarrow \text{In most applications, we additionally require zero-knowledge.} \]

Note: knowledge is a strictly stronger property than soundness

\[ \rightarrow \text{if protocol has knowledge error } \varepsilon \Rightarrow \text{it also has soundness error } \varepsilon \text{ (i.e., a dishonest prover convinces an honest verifier of a false statement with probability at most } \varepsilon) \]

**Proving knowledge of discrete log (Schonherr's protocol)**

Suppose prover wants to prove it knows \( x \) such that \( h = g^x \) (i.e., prover demonstrates knowledge of discrete log of \( h \) base \( g \))

```
prover
```

```
v \leftarrow g^r
```

```
\text{verify that } g^2 = u \cdot h^c
```

```
verify that \text{knowledge error is negligible}
```

```
assume g \in G \text{ where } G \text{ has prime order } q
```

```
\rightarrow \text{efficient extractor } E
```

```
\rightarrow \text{interactive proof system } (P, V)
```

```
\rightarrow \text{proves knowledge of discrete log}
```

```
\rightarrow \text{security against dishonest provers}
```

```
\rightarrow \text{zero-knowledge property}
```

```
\rightarrow \text{acquires no additional information from verifier}
```

```
\rightarrow \text{advantage of honest verifier is negligible}
```

```
\rightarrow \text{amplifiable to any polynomial number of rounds}
```

```
\rightarrow \text{proven in the 1980s}
```

```
\rightarrow \text{black-box use of non-interactive proofs}
```

```
\rightarrow \text{efficiently extractable knowledge}
```

```
\rightarrow \text{interactive zero-knowledge proofs}
```

```
\rightarrow \text{introduced by Schnorr}
```

```
\rightarrow \text{provably secure against malicious provers}
```

```
\rightarrow \text{used in many protocols}
```

```
\rightarrow \text{still popular today}
```
Completeness: if \( z = r + cx \), then 
\[
    g^z = g^{r + cx} = g^r g^{cx} = u \cdot h^c
\]
zero knowledge only required to hold against an honest verifier
(e.g., view of the honest verifier can be simulated)

Honest-Verifier Zero-Knowledge: build a simulator as follows (similar strategy: run the protocol in reverse):

on input \((g, h)\):
1. Sample \( z \overset{\$}{\leftarrow} \mathbb{Z}_p \)
2. Sample \( c \overset{\$}{\leftarrow} \mathbb{Z}_p \)
3. Set \( u = g^z / h^c \) and output \((u, c, z)\)

Simulated transcript is identical to distributed as the real transcript with an honest verifier.

What goes wrong if the challenge is not sampled uniformly at random (i.e., if the verifier is dishonest)?
Above simulation no longer works (since we cannot sample \( z \) first)

\( \Rightarrow \) To get general zero knowledge, we require that the verifier first commit to its challenge (using a statistically hiding commitment)

Knowledge: Suppose \( P^* \) is (possibly malicious) prover that convinces honest verifier with probability 1. We construct an extractor as follows:

1. Run the prover \( P^* \) to obtain an initial message \( u_1 \).
2. Send a challenge \( c_1 \overset{\$}{\leftarrow} \mathbb{Z}_p \) to \( P^* \). The prover replies with a response \( z_1 \).
3. "Replay" the prover \( P^* \) so its internal state is the same as it was at the end of Step 1. Then, send another challenge \( c_2 \overset{\$}{\leftarrow} \mathbb{Z}_p \) to \( P^* \). Let \( z_2 \) be the response of \( P^* \).
4. Compute and output \( \chi = (z_1 - z_2)(c_1 - c_2)^{-1} \in \mathbb{Z}_p \).

Since \( P^* \) succeeds with probability 1 and the extractor perfectly simulates the honest verifier's behavior, with probability 1, both \((u, c_1, z_1)\) and \((u, c_2, z_2)\) are both accepting transcripts. This means that
\[
    g^z = u \cdot h^c \quad \text{and} \quad g^{z_2} = u \cdot h^{c_2}
\]
\[
\Rightarrow g^{z_1} = g^{z_2} / h^{c_2} = g^{z_1 + c_2 \chi} \quad \text{with overwhelming probability,}
\]
\[
\Rightarrow \chi = (z_1 - z_2)(c_1 - c_2)^{-1} \in \mathbb{Z}_p \quad c_1 \neq c_2
\]

Thus, extractor succeeds with overwhelming probability.

(Boaz-Shoup, Lemma 19.2)

If \( P^* \) succeeds with probability \( \varepsilon \), then need to rely on "Reckoning Lemma" to argue that extractor obtains two accepting transcripts with probability at least \( \varepsilon^2 - \varepsilon / p \).

The ability to extract a witness from any two accepting transcripts is very useful.

\( \Rightarrow \) called special soundness (for 3-message protocols)

given \((u, t, z_1)\) and \((u, t, z_2)\) \( \Rightarrow \) can extract the witness

\[\text{initial message} \downarrow \text{challenge} \uparrow \text{response} \quad \text{[same initial message, different challenges]}\]