We will need a commitment scheme (see HWS). A (non-interactive) commitment scheme consists of two main algorithms (Commit, Verify):

- **Commit** \( (m, r) \rightarrow c \) : Takes a message \( m \) and randomness \( r \) and outputs a commitment \( c \)
- **Verify** \( (m, c, r) \rightarrow b \) : Checks if \( c \) is a valid opening to \( m \) (with respect to randomness \( r \))

[The commitment scheme might also take public parameters (see HWS), but for simplicity, we omit them here or make them implicit]

**Requirements:**

- **Correctness:** for all messages \( m \): sampled uniformly
  \[ \Pr[c \leftarrow \text{Commit}(m; r) : \text{Verify}(m, c, r) = 1] = 1 \]

- **Binding:** for all efficient adversaries \( A \), if \( (m, r) \leftarrow A \)
  \[ \{c \leftarrow \text{Commit}(m; r) : c \} \approx \{c \leftarrow \text{Commit}(m'; r) : c \} \]

- **Hiding:** for all efficient adversaries \( A \), \( (m, c, r) \leftarrow A \) \[ \text{if } \Pr[(m', c, r) \leftarrow A : m' \neq m \text{ and } \text{Verify}(m', c, r) = 1 = \text{Verify}(m, c, r) ] = \text{negl} \]

\( \Rightarrow \) **We will require perfect binding** [for every commitment \( c \), there is only 1 possible \( m \) to which the prover can open \( c \)]

A 2K protocol for graph 3-coloring:

![Diagram](https://example.com/diagram.png)

- Prover \( (G) \) contains \( n \) nodes, \( m \) edges

- Let \( K_i \in \{0,1,2\} \) be a 3-coloring of \( G \)

- Choose random permutation \( \Pi \) of \( \text{Perm}([n]) \)

- for \( i \in [n] \):
  \[ c_i \leftarrow \text{Commit}(K_i; r) \]

- for random \( r \):
  \[ c_1, \ldots, c_n \rightarrow (i,j) \in E \]
  \[ \leftarrow (K_i, r_i), (K_j, r_j) \]

\( \Rightarrow \) accept if \( k_i \neq k_j \) and \( k_i, k_j \in \{0,1,2\} \)

\[ \text{Verify}(k_i, c_i, r_i) = 1 = \text{Verify}(k_j, c_j, r_j) \]

reject otherwise

**Intuition:** 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]

**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 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 \( \frac{1}{|E|} \). 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

\[ \Pr[\text{verifier accepts proof of false statement}] \leq (1 - \frac{1}{|E|})^{|E|^2} \leq e^{-|E|} = e^{-n^2} \quad [\text{since } 1 + x \leq e^x] \]
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) simulator. Construct simulator $S$ as follows:

1. Choose $K_i \leftarrow \{0,1,2,3\}$ for all $i \in [n]$. Let $C_i \leftarrow \text{Commit}(K_i; r_i)$
   Give $(C_1, \ldots, C_n)$ to $V^*$.

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

Simulator succeeds with probability $2/3$ (over choice of $K_i,\ldots,K_n$). 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 simulate one transcript at a time.

Summary: Every language in $NP$ has a zero-knowledge proof.

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:

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

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 $P'$

$$
Pr[w \leftarrow E^{P'}(x) : R(x, w) = 1] \geq Pr[P(x) = 1] - \epsilon.
$$

more generally, could be polynomially smaller

$\epsilon$ could be polynomially smaller

proof of knowledge is parameterized by a specific relation $R$ (as opposed to the language $L$)
Trivial proof of knowledge: prover sends witness in the clear to the verifier.

- In most applications, we additionally require zero-knowledge.

Note: knowledge is a strictly stronger property than soundness.

- If protocol has knowledge error $\varepsilon \Rightarrow$ it also has soundness error $\varepsilon$ (i.e., a dishonest prover convinces an honest verifier of a false statement with probability at most $\varepsilon$).

Proving knowledge of discrete log (Schnorr'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 $g$).

\[
\begin{array}{c|c|c}
\text{prover} & \text{verifier} \\
\hline
r \in \mathbb{Z}_p^* & u \leftarrow g^r & c \leftarrow z \\
\hline
z \leftarrow r + cx & g^z = g^{r+cx} = g^r g^{cx} = u \cdot h \\
\end{array}
\]

Completeness: if $z = r + cx$, then $g^z = g^{r+cx}$.

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.

1. Sample $z \in \mathbb{Z}_p^n$.
2. Sample $c \in \mathbb{Z}_p^n$.
3. Set $u = g^z / h^c$ and output $(u, c, z)$.

\text{Simulated transcript is identically 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).

As 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 $u$.
2. Send a challenge $c_1 \in \mathbb{Z}_p^n$ to $P^*$. The prover replies with a response $z_1$.
3. "Rewind" the prover $P^*$ so its internal state is the same as it was at the end of Step 1. Then, send another challenge $c_2 \in \mathbb{Z}_p^n$ to $P^*$. Let $z_2$ be the response of $P^*$.
4. Compute and output $x = (z_1 - z_2)(c_1 - c_2) \in \mathbb{Z}_p$. 

For simplicity, we assume $P^*$ succeeds with probability $\varepsilon$. "$x$ is the secret.\]
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
\[
\frac{g^{z_1}}{h^{c_1}} = u \cdot h^c \quad \text{and} \quad \frac{g^{z_2}}{h^{c_2}} = u \cdot h^c
\]
\[\Rightarrow \frac{g^{z_1}}{h^{c_1}} = \frac{g^{z_2}}{h^{c_2}} \quad \Rightarrow \quad g^{z_1 - z_2} = \frac{z_1 + c_1 x}{z_2 + c_2 x}
\]
\[\Rightarrow x = (z_1 - z_2)(c_1 - c_2)^{-1} \in \mathbb{Z}_p \quad c_1 \neq c_2.
\]
Thus, extractor succeeds with overwhelming probability.

(Boneh-Shoup, lemma 19.2)

If $P^*$ succeeds with probability $E$, then need to rely on "Rewinding Lemma" to argue that extractor obtains two accepting transcripts with probability at least $E^2 - \frac{1}{p}$.

How can a prover both prove knowledge and yet be zero-knowledge at the same time?

1. Extractor operates by "rewinding" the prover. If the prover has good success probability, it can answer most challenges correctly.
2. But in the real (actual) protocol, verifier cannot rewind (i.e., verifier only sees prover on fresh protocol executions), which cannot provide zero-knowledge.