Today:
- Finish definition of ZKPoK
- ZK for more languages "statement"

\[ \text{ZKPoK}_x \in (w) : L \left( \begin{array}{c} x, w \end{array} \right) \]

Languages
a language \( L \) is a set
\[ x \in L \]
\[ x \in L \text{ is an NP-language:} \]
\[ \forall x \in L \]

Defn:
A ZKPoK scheme for language \( L \) is a PPT Prover \( (P) \) and Verifier \( (V) \) satisfying:
- Correctness: \( \forall x \in \Sigma \; w \in W, \; L \left( \begin{array}{c} x, w \end{array} \right) = 1 \);
\[ \text{Output} \left[ P(x, w) \leftrightarrow V(x) \right] = 1 \]
- Honest Verifier Zero-knowledge"
"Simulatability":
\[ \exists S, \; \exists \text{View}[P(x, w) \leftrightarrow V(x)] \]
\[ \forall \xi, \; \exists S(x) \]

Crypto Jake
Fire Distinguisher

- Extractable:
\[ \forall A, \; \text{Pr} \left[ \text{Output} \left[ A(x) \leftrightarrow V(x) \right] = 1 \right] = \text{negl.} \]

Then \( \exists E \), such that \( \text{Pr} \left[ w \leftarrow E_A(x) : L \left( \begin{array}{c} x, w \end{array} \right) = 1 \right] = \text{negl.} \)

For the Schnorr protocol:
\[ x, u, y \sim \]
\[ c \sim \]
- Correctness:
  $$\delta = \delta - (g)(h') - m \cdot n$$

- Simulation:
  $$S(x) = c, c \in Z_p \setminus \{0\}, s \in Z_{161}, K = \delta / X$$

- Extractor:
  $$E_{\delta}(x)$$

  Suppose
  $$A \leftarrow PF(A' \leftarrow N)$$ output

  $$A, K \rightarrow v$$

  Define:

  Run $$A(x)$$ until it outputs $$K$$.
  Make a "snapshot" of $$A$$ as $$A'$$.
  Sample
  $$c_1 \leftarrow Z_{161} \setminus \{0\}$$
  $$c_2 \leftarrow Z_{161} \setminus \{0\}$$

  Let
  $$s_1 = A'(c_1)$$
  $$s_2 = A'(c_2)$$

  Note that with $$p = X$$
  $$X \cdot K = \delta$$

  Check this, repeat as necessary!

We solve for
  $$\overline{x} = (s_1 - s_2) / (c_1 - c_2)$$

  Extended euclidean algorithm

  $$p = 6$$

  $$x = \overline{x}$$

  Comp. Sound: $$x \in L$$, Prover doesn't necessarily "know" $$w$$

**Extending ZK Proofs to Other Languages:**

- ZK Proofs are secure:
  $$\delta = A, \delta = B$$

  - Repeat with twice?
  - Use the same $$c$$?

  $$P (a, b)
  k_1 \leftarrow Z_p$$
  $$k_2 \leftarrow Z_p$$

  $$\overline{K_1, K_2} \rightarrow c \leftarrow Z_p \setminus \{0\}$$

  $$S_1 = k_1 + ca$$
  $$S_2 = k_2 + cb$$

  $$S_1, S_2 \rightarrow \text{Check} \overline{S_1, S_2} \leftarrow \overline{A^c K_1}$$
To check:
- correctness
- simulatability
- extractability

$\text{View}_V(\{K_1, K_2, \xi, s_1, s_2\})$

**Commitments:**

$\text{(com, open)}$

- hiding $\text{com}_r(x) \rightarrow c$ reveal nothing about $x$ for
  
  $r \in Z_p$

- binding Cannot generate collision $(r, r', x, x', c)$ s.t.
  $\text{open}(r, x, c) = 1$
  $\text{open}(r', x', c) = 1$

**Pedersen Commitment:**

Uses $h \in G$ (an alternate generator)

$\text{Com}_r(x) = g^x h^r$ hiding

$\text{open}(r, x, c)$: check $c = g^x h^r$

**Hiding:**

Given $x$, bijective from $r$ to $c$.

$\delta_x(r) = g^x h^r$

$\text{h} = g^x$

$g^x g^r = g^{x+r}$

**Binding:**

**Reduction to Discrete Log**

New proof technique!!

Suppose $A$ is

$\text{Adv}^\text{Binding}_A = \Pr[\text{com}_r(x, s_1, x, x) \leftarrow A(m) \text{ and } \text{open}(r, x, c) = 1, \text{open}(r, x, c) = 1]$

Then, we construct $A'$ that wins DLOG

$A'(X) \leftarrow \text{group elem}$

$\text{goal: output } x \text{ s.t. } X = g^x$

Let $h = X$

(recur)
\( (c, r_1, r_2, x_1, x_2) \subseteq A(l, u) \)

\[ c = g^{x_1}h = g^{x_2}h \]

Solve:

\[ x = (x_1 - x_2)/(r_2 - r_1) \]

\[ x = g^{x_1}h \]

\[ x = g^{x_2}h \]

Output \( x \). Solution

\( \exists (x, r) \subseteq (x, r) : C = g^{x_1}h \leq 3 \)