Zero Knowledge Proofs notation

- Alice \( x \) \( g^x = X \)
  - private key
  - public key

ZKP is
- Alice can say
  - statement
  - relation/predicate

"I know my secret key \( x \) s.t.
\( X = g^x \)"

v/o revealing \( x \)

\[ \text{ZKP} \{ (x); X = g^x \} \]

- witness
- statement
- relation/predicate

Where's Wildo ZKP.

Language \( L_{	ext{ENS}} \) is a set of strings

\[ x \in L \iff \exists w, s.t. L(x, w) = 1 \]

Ideal Functionality

Verifier

Prover

Alice \( x, w \)

Bob

"I learn \( x \in L \) for free!"

"protocol is secure iff it is as good as the IF"
Interactive Protocol

\[
\begin{array}{c}
\text{P (X,w)} \\ \text{prover starts with stmt}
\end{array} \quad \xrightarrow{\text{"challenge"}} \quad \begin{array}{c}
\text{V(X)} \\ \text{verifier starts with just statement}
\end{array} \\
\downarrow \quad \downarrow
\begin{array}{c}
\text{output accept/reject}
\end{array}
\]

\[ - \text{out}_V[\{P(x,w) \leftrightarrow V(x)\}] \]
\[ \text{means output of } V \]

\[ - \text{view}_V[\{P(x,w) \leftrightarrow V(x)\}] \]
\[ \text{means a transcript of all messages received by } V \]
\[ \text{random choices made by } V. \]

**Correctness:** "Verifier accepts if prover is honest"

\[
\forall X, w. \ L(x,w) = 1, \quad P_r \left[ \text{out}_V[P(x,w) \leftrightarrow V(x)] = 1 \right] = 2
\]

**Soundness:** "Verifier only accepts X \in L"

\[
\forall A, X. \quad P_r \left[ \text{out}_V[A(x) \leftrightarrow V(x)] = 1 \quad \text{and} \quad X \notin L \right] \leq Req
\]

**Extraction** (Stronger than soundness)

"Verifier only accepts if X \in L and prover "knows" witness w"

**Zero Knowledge:** "Verifier knows no more information after the protocol than before"
"View of the verifier can be simulated even without interacting with the prover."

\[ \forall X, v, L(X), \exists S \text{ simulator} \]

\[ \text{View}_v[P(X,v) \leftrightarrow V(X)] \approx \leq (X) \]

\[ \text{real transcript initiating of prover.} \]

\[ \text{Simulated transcript} \]

**Protocol for** \( 2 \text{V} \leq (X); X = g^x \) \( |G| = p \)

\[ P(X;X): \]

\[ k \in \mathbb{Z}_p \]

\[ K = g^k \]

\[ \text{"commit"} \]

\[ \text{"challenge"} \]

\[ c \in \mathbb{Z}_p \]

\[ S = xc + k \]

\[ \text{"response"} \]

\[ g^s = X^cK \]

- **Correctness:**

  \[ g^s = g^{xc+k} = (g^x)^c g^k = X^cK \]

- **Zero-knowledge:**

  \[ \text{View}_v[P(X,v) \leftrightarrow V(X)] \]

  \[ \begin{cases} \frac{1}{p^2} & \text{if } g^s = X^cK \\ 0 & \text{otherwise} \end{cases} \]
\[ S(X) : K \in \mathbb{G}_1, c \in \mathbb{Z}_p, s = \log_2 K + c \log_2 X \]
\[ c \in \mathbb{Z}_p, s \in \mathbb{Z}_p, K = s^c / x^c \]

- Extraction for next time:
  - If \( A \) produces a valid proof for \( X \), with high probability of extractability, then we can use \( A \) to adapt a witness for \( \mathcal{G} \) (run \( A \) multiple times).

\[
\begin{array}{ccc}
A_{(1^2, x; Z)} & c \in \mathcal{G}_1 & \Rightarrow \ A \text{ adapts witness} \\
A_{(1^2, x; Z)} & \text{random} & \text{random} \\
K \rightarrow c \rightarrow & E_{c} & E_{c} \\
\end{array}
\]

We construct an extractor \( E_{\mathcal{A}}(1^2, X) : \)
\[ Z \in \mathcal{G}, 1^2 \]
\[ \text{run} \ A \text{ until it outputs first message } K, \]
\[ A_{(1^2, x; Z)} \rightarrow K \]

\[ c_1 \in \mathbb{Z}_p \]
\[ \text{Send } c_1 \text{ to } A, \text{ run until receiving } s_1. \]
run $\text{UA}(1^n, X; Z)$ a second time... and adapt $K$.

(2) \[ c_2 \in \mathbb{Z}_p \]

Send $c_2$ to $U$, receive $s_2$.

If $\text{adv}[A^{c_2}] = 1$ with prob $\frac{1}{d}$

then \[ g^{s_1} = X^{c_2} K \]
and \[ g^{s_2} = X^{c_2} K \]

$g^{s_1}/g^{s_2} = X^{c_2} K / X^{c_2} K$

$g^{s_1-s_2} = X^{c_1-c_2}$

$s_{12} \left[ \begin{array}{c} g^{s_1-s_2} \\ (X^{c_1-c_2}) \end{array} \right] \left[ \begin{array}{c} c_1-c_2 \end{array} \right] \left[ \begin{array}{c} X^{c_1-c_2} \end{array} \right]$

since $g = (s_1-s_2)/(c_1-c_2)$.

Commitments:

Create $C$ with a protocol for

$Z \cup \{ (Xr) : C = g^{Xr} \}$