Today

- Small ZK Recap
- Proofs of Knowledge
- Schnorr Protocol
- Sigma Protocols
- Variants (AND/OR)
Recap—Zero Knowledge Proofs

Let $L \subseteq \{0,1\}^*$ be an NP-language.

A ZK Proof System is a tuple of efficient interactive alg's $(P, V)$ s.t. they satisfy

Properties

1) Completeness: $\forall x \in L$, $\Pr[<P(x,w), V(x)>=1] = 1$  
2) Soundness: $\forall x \not\in L, \forall P^*$, $\Pr[<P^*, V(x)>=1] \leq \text{neg}(\lambda)$  
3) ZK: $\exists \text{ PPT Sim}, \forall \text{ PPT } V^*$,  
   $(\exists \text{ View}_{V^*}(P(x,w), V^*>) \approx (\exists \text{ Sim}^*(x)>)$  
   “malicious verifier ZK”

Honest Verifier ZK (HVZK): $\exists \text{ PPT Sim},$  
$(\exists \text{ View}_{V}(P(x,w), V>) \approx (\exists \text{ Sim}(x)>)$  
\hspace{2cm} honest verifier \hspace{2cm} no oracle access

Proofs of Knowledge:

Soundness (informally): the verifier is convinced that $x$ (a graph $G$) is in a language $L$ (HAM CYCLE Graphs).

However, in many cases, we want to verify that a prover actually “knows” a witness (a HAM CYCLE).

We would like a Proof of Knowledge (Pok).
i.e. if $V$ accepts w.h.p., then $P$ must know a witness $w$.

Does the soundness property imply a proof of knowledge? **No!** Consider the following NP-Relation,

$$R_{\text{composite}} = \{ (N, p) \mid p \mid N \land p \notin \{1, N^2\} \}$$

- Verifying a number is not prime is not the same as factoring

**Intuition:** How can we guarantee that a malicious prover $P^*$ knows a $w$ s.t. $(X, w) \in R$?

**A trivial Pok (attempt 1):**

\[ \begin{align*}
  \text{P}(X, w) & \quad \text{W} \\
  \quad \text{V}(X) & \quad \text{Check } (X, w) \in R? \\
\end{align*} \]

**Issue:** cannot be EK! since need to simulate an actual witness.

**Attempt 2**

\[ \begin{align*}
  \text{P}(X, w) & \quad \text{V}(X) \\
  \quad \text{mi} & \quad \text{Check } (X, w) \in R? \\
\end{align*} \]

Given messages $\epsilon mi$ from $P^*$, we can compute a satisfying witnesses... SAME Issue!
What if instead we can interrogate the prover multiple times?

Let $E$ be a ppt alg called an extractor

![Diagram of E](image)

$E$ has black box access to $P^*$, can interact, "rewind" it to previous rounds.

**What does rewinding mean?**

An interactive ppt alg $P^*$ can be described as a series of next message functions: [BG92]

Let $p \leftarrow \{0,1\}^*$ represent the prover's private randomness

$p^*$

\[
\begin{array}{c}
\leftarrow m_1 \\
r_i \\
\vdots
\end{array} \quad \begin{array}{c}
V(x) \\
m_n \\
r_n \\
\end{array} \quad \begin{array}{c}
p_1^* (p, m_i) \rightarrow r_1 \\
p_2^* (p, m_i, m_2) \rightarrow r_2 \\
\vdots \\
p_n^* (p, m_i, \ldots, m_n) \rightarrow r_n \\
\end{array}
\]

We define $E_p^*$ as the extractor that has oracle access to the functions $E_{P_i^* (p, \cdot)}$. Thus, $E$ can "rewind" $P^*$. 
\((P, V)\) is a Pok for NP relation \(R\) with knowledge error \(K\) if
\[
\exists \text{ PPT } \exists \text{ s.t. } \forall x, \forall \rho^*.
\]
\[
Pr[(x, w) \in R : w \in \mathcal{E}^p(x)] \geq Pr[<\rho^*, V(x)> = 1] - K
\]

\textbf{Schnorr's Protocol}

Let \(G\) be a group of prime order \(q\) with a generator \(g\). Define \(R_{\text{dlog}} = \mathcal{E}(h, x) \mid h = g^x \). Note \(L(R) = G\) is a trivial language. Thus, soundness is a trivial property to satisfy, but is Pok?

Prover wants to convince \(V(h)\) that it knows the discrete log of \(h\).

**Prover**

\[
\begin{align*}
    &P(x \in \mathbb{Z}_q, h = g^x \in G) \\
    &r \leftarrow \mathbb{Z}_q \\
    &u = g^r \\
    &c \leftarrow \mathbb{Z}_q
\end{align*}
\]

\[
\begin{align*}
    &z = r + cx \\
    &g^z = g^{r + cx} = g^r (g^x)^c = u \cdot h^c
\end{align*}
\]

\[
\begin{align*}
    &V(h) \\
    &\leftarrow \text{“challenge”} \\
    &\text{Output } g^z = u \cdot h^c
\end{align*}
\]

**Claim:** Schnorr's Protocol is an honest-verifier ZK-PoK of \(\text{DLog}\).

Completeness

\[
g^z = g^{r + cx} = g^r (g^x)^c = u \cdot h^c
\]
Simulator runs the protocol in "reverse":
\[ \text{Sim}(h) \]

1) Sample \( z \leftarrow \mathbb{Z}_q \)
2) Sample \( c \leftarrow \mathbb{Z}_q \)
3) Set \( u := g^{z/c} \)
4) Output \((u, c, z)\)

\( g^z = u \cdot h^c \) (transcript satisfies verifier check)

Can we get malicious \( ZK \)?

Issue: malicious verifiers challenge \( c \) may not be uniform random so strategy above of sampling \( z \) first no longer works.

Folklore Result: to get full \( ZK \), have the verifier commit to their challenge before seeing \( u \).

[Lindell: errata-ZK-sigma]
Proof of Knowledge

Suppose $p^*$ convinces an honest verifier $V(h)$ with probability $\epsilon = 1$.

Intuition: Let us rewind the prover to operate on different challenges

\[\begin{align*}
(1) & \quad p^* \quad V(h) \\
& \quad u \\
& \quad c_i \leftarrow Z_q \\
& \quad z, \\
& \quad g^{z_i} = u \cdot h^{c_i} \\
(2) & \quad p^* \quad V(h) \\
& \quad u \\
& \quad c_z \leftarrow Z_q \\
& \quad z, \\
& \quad g^{z_z} = u \cdot h^{c_z}
\end{align*}\]

Since we assumed $\epsilon = 1$, then $(u, c_i, z_i)$ and $(u, c_z, z_z)$ are two accepting transcripts. Thus, $g^{z_i} = u \cdot h^{c_i}$ and $g^{z_z} = u \cdot h^{c_z}$

\[g^{z_i - z_z} = h^{c_i - c_z} \]

W.h.p. $c_i \neq c_z$,

\[g^{\frac{z_i - z_z}{c_i - c_z}} = h \quad \Rightarrow \quad x = \frac{z_i - z_z}{c_i - c_z} \text{ is the DLog of } h.\]

More formally

\[E_{p^*} \]

1) Run $p^*$ to get $u$.
2) Send $c_i \leftarrow Z_q$, and receive $z_i$.
3) Rewind $p^*$, send $c_z \leftarrow Z_q$, and receive $z_z$.
4) If $c_i = c_z$, output fail. O/W output $x = \frac{z_i - z_z}{c_i - c_z}$. 
Analysis

\[ \Pr[(h,x) \in \mathcal{R}_{\text{Env}} : x \leftarrow E^*(h)] = 1 - \frac{1}{q} \geq \Pr[\langle \rho^*, \nu \rangle(h) = 1] - \frac{1}{q} \]

Thus, \( K = \frac{1}{q} \).

We assumed \( \Pr[\langle \rho^*, \nu \rangle(h) = 1] = 1 \), but more generally what about \( \Pr = \varepsilon \)?

**Rewinding Lemma (BS 19.2)**

If \( \rho^* \) succeeds with probability \( \varepsilon \), then using the "rewinding lemma", we can argue the extractor obtains two accepting transcripts (with \( c_1 \neq c_2 \)) with prob at least \( \varepsilon^2 - \varepsilon^2 q \).

**Sigma Protocols (Z-Protocols)**

More broadly, the Schnorr Protocol belongs to a family of three message protocols called Sigma Protocols.

\[
\begin{array}{c}
P(x, w) \\
\hline
\text{t "commitment"} \\
\hline
\text{c "challenge"} \\
\hline
\text{z "response"} \\
\hline
V(x)
\end{array}
\]

\( c \leftarrow \mathcal{C} \)

Output 0/1 deterministically from \((x, t, c, z)\)
Properties:

1) Perfect Completeness
deterministic

2) Special Soundness: \exists\text{ extractor } E \text{ that given two accepting transcripts } (t, c, z), (t, c', z') \text{ with } c \neq c' \text{ outputs } w \text{ s.t. } (x, w) \in R.

\Rightarrow \text{ PoK (can you see how?)}

3) Special Honest Verifier EK: \exists efficient Sim(x, c) \rightarrow (t, z) \text{ s.t. } (t, c, z) \text{ is an accepting transcript for } x.

Additionally, \forall (x, w) \in R,

\{ (t, c, z) : c \leftarrow c \leftarrow \text{Sim}(x, c) \} \approx \{ \text{View}_v(P(x, w), V(x)) \}

* In literature, you may see \mu-round Sigma Protocols with (k_1, k_2, \ldots, k_\mu) - special soundness. The Schnorr Protocol is a 1-round Sigma Protocol with 2-special soundness referring to number of distinct challenges needed at round i ≤ \mu.

**AND Proofs**

Let (P_0, V_0) and (P_1, V_1) be Sigma Protocols for relations R_0 and R_1 respectively that use the same challenge space C. Define the following AND-Relation

\[ R_{\text{and}} := \{(x_0, x_1) ; (w_0, w_1) \mid (x_0, w_0) \in R_0 \land (x_1, w_1) \in R_1 \} \]
We can construct a Sigma Protocol \( (P,V) \) for \( Rand \) as follows:

\[
\begin{align*}
P((x_0,x_1),(w_0,w_1)) & \\
\text{Run } & P_0(x_0,w_0) \rightarrow t_0 \\
P_1(x_1,w_1) \rightarrow t_1 & \\
\text{Feed } c \text{ to } P_0, P_1 & \\
\text{to obtain } (z_0,z_1) & \\

V((x_0,x_1)) & \\
(t_0,t_1) & \\
\text{Accept if both } & \\
V_0(t_0,c,z_0) & \text{ and } \\
V_1(t_1,c,z_1) & \text{ accept }
\end{align*}
\]

**Proof Sketch**

**Special Soundness:**

Given two accepting transcripts \( c \neq c' \):

\[
((t_0,t_1),c,(z_0,z_1)),((t_0,t_1),c',(z_0',z_1'))
\]

Run extractors \( E_0((to,c,z_0),(to,c',z_0')) \rightarrow w_0 \)

\( E_1((t_1,c,z_1),(t_1,c',z_1')) \rightarrow w_1 \)

Output \( (w_0,w_1) \)

**HVZK**

\[
\begin{align*}
\text{Sim } ((x_0,x_1),c) : \\
(t_0,z_0) & \leftarrow \text{Sim}_0(x_0,c), (t_1,z_1) \leftarrow \text{Sim}_1(x_1,c) \\
\text{Output } ((t_0,t_1),c,(z_0,z_1)) & \\
\approx \text{Only a Sketch: need to argue extractor is correct } & \\
\text{and simulator distribution is indistinguishable}
\end{align*}
\]
OR - Proof

\[ R_{OR} := \{ (x_0, x_1), (b \in \{0, 1\}, w_b) \mid (x_b, w_b) \in R_b \} \]

\[ P(x_0, x_1, b, w_b) \]

- Compute \( c_b \leftarrow C \)
- \((t_b, z_b) \leftarrow \text{Sim}_b(x_b, c_b)\)
- Run \( P_b(x_b, w_b) \rightarrow t_b \)

\[ \text{Compute } c_b \leftarrow c \oplus c_b \]
- Feed \( c_b \) to \( P_b \) to get \( z_b \)

- Assume \( C \) has an XOR \( \oplus \) operation

Proof Sketch

Special Soundness

Given \((t_0, t_1), c, (c_0, z_0, z_1), (t_0, t_1), c', (c_0', z_0', z_1')\).

Define \( c_i := c \oplus c_0 \) and \( c_i' := c' \oplus c_0' \). Notice since \( c' \neq c \), then either \( c_0 \neq c_0' \) or \( c_i \neq c_i' \).

If \( c_0 \neq c_0' \):

Output \((0, E_0(x_0, (t_0, c_0, z_0), (t_0, c_0', z_0')))\)

Else:

Output \((1, E_1(x_1, (t_1, c_i, z_1), (t_1, c_i', z_i')))\)
\[ H_{\text{VZK}} \]

\[ \text{Sim}((x_0, x_1), c) \]
- \( c_0 \in C, c_1 \in C \oplus c_0 \)
- \((t_0, z_0) \in \text{Sim}_o(x_0, c_0) \)
- \((t_1, z_1) \in \text{Sim}_i(x_1, c_1) \)

Output \(( (t_0, t_1), C, (c_0, z_0, z_1) ) \)

**Summary**

Today, we learned

- What are Proofs of Knowledge
- Example of Pok is Schnorr Protocol (Proof of Dlog)
- Schnorr Protocol belongs to Sigma Protocols
- AND/OR Prots for combining Sigma Protocols