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 algs (P,V) s.t. they satisfy

Properties

sometimes $\geq 1 - negl(\lambda)$

1) <u>Completeness:</u> $\forall x \in L$, $Pr[\langle P(x,w), V(x)\rangle = 1] = 1$
2) <u>Soundness:</u> $\forall x \notin L$, $\forall P^*$, $Pr[\langle P^*, V(x)\rangle = 1] \leq negl(\lambda)$
3) <u>ZK:</u> $\exists$ PPT Sim, $\forall$ PPT V^*,
   $\{View_{V^*}(\langle P(x,w), V^*\rangle)\} \approx \{Sim^{V^*}(x)\}$
   "malicious verifier ZK"

<u>Honest Verifier ZK (HVZK):</u> $\exists$ PPT Sim,

$$\{View_V(\langle P(x,w), V\rangle)\} \approx \{Sim(x)\}$$

honest verifier

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,

$$R_{\text{Composite}} = \{(N; p) \mid p \mid N \wedge p \notin \{1, N\}\}$$

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

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

A trivial PoK (attempt 1):

$P(x, w)$		$V(x)$
	$\xrightarrow{w}$	Check $(x,w) \in R$?
		(eff since R is an NP relation)

<u>Issue:</u> This cannot be ZK, as the simulator would have to efficiently output a valid witness.

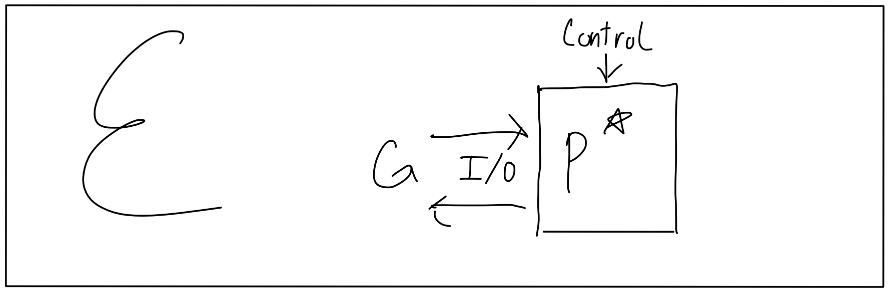
<u>Attempt 2</u>

$P(x, w)$		$V(x)$
	$\xrightarrow{m_i}$	
	$\xleftarrow{r_i}$	

Given messages $\{m_i\}$ from P^*, we can compute a satisfying witness ... SAME Issue!

What if instead we can interrogate the prover multiple times?

Let $\mathcal{E}$ be a PPT alg called an extractor

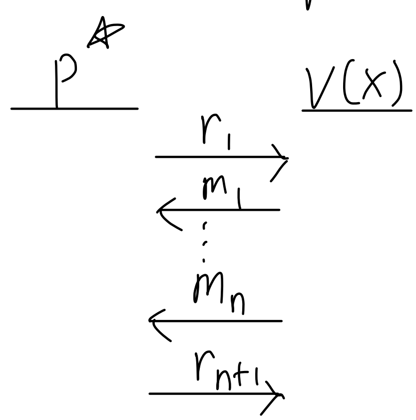


$\mathcal{E}$ has black box access to P^*, can interact, "rewind" it to previous rounds.

What does rewinding mean?

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

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



$\iff$

$$P_1^*(\rho) \to r_1$$
$$P_2^*(\rho, m_1) \to r_2$$
$$\vdots$$
$$P_n^*(\rho, m_1, \dots, m_n) \to r_{n+1}$$

We define $\mathcal{E}^{P^*}$ as the extractor that has oracle access to the functions $\{P_i^*(\rho, \cdot)\}_i$; Furthermore, we allow the extractor to force a resampling of the initial prover randomness ρ.

(P, V) is a PoK for NP relation R with knowledge error κ if
$\exists$ PPT $\mathcal{E}$ s.t. $\forall x, \forall P^*$,

Sometimes expected poly time

$$\Pr\left[(x,w) \in R : w \leftarrow \mathcal{E}^{P^*}(x)\right] \geq \Pr\left[\langle P^*, V(x)\rangle = 1\right] - \kappa$$

Sometimes polynomially smaller (include a slack)

Schnorr's Protocol

Let $\mathbb{G}$ be a group of prime order q with a generator g.
Define $R_{DLog} = \{(h, x) \mid h = g^x\}$. Note $L(R) = \mathbb{G}$ 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.

$P(x \in \mathbb{Z}_q, h = g^x \in \mathbb{G})$ — $V(h)$

$r \xleftarrow{\$} \mathbb{Z}_q$

$P \xrightarrow{u = g^r} V$

$c \xleftarrow{\$} \mathbb{Z}_q$ ("challenge")

$P \xleftarrow{c} V$

$P \xrightarrow{z = r + cx} V$

Output $g^z \stackrel{?}{=} u \cdot h^c$

Claim: Schnorr's Protocol is an honest-verifier ZK-PoK of DLog.

Completeness

$$g^z = g^{r+cx} = g^r (g^x)^c = u \cdot h^c$$

HVZK:

Simulator runs the protocol in "reverse":

Sim(h)

1) sample $z \xleftarrow{\$} \mathbb{Z}_q$
2) sample $c \xleftarrow{\$} \mathbb{Z}_q$
3) Set $u := g^z / h^c$
4) Output (u, c, z)

uniform random

uniform random since c is uniform

uniquely determined by u and c

$g^z = u \cdot h^c$

(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.

Folk Lore Result: to get full ZK, have the verifier commit to their challenge before seeing u. However, this introduces an additional round of communication.

[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

(1)

$$\begin{array}{ccc}
\underline{P^*} & & \underline{V(h)} \\
 & \xrightarrow{\;u\;} & \\
 & \xleftarrow{\;c_1 \xleftarrow{\$} \mathbb{Z}_q\;} & \\
 & \xrightarrow{\;z_1\;} & g^{z_1} \stackrel{?}{=} u \cdot h^{c_1}
\end{array}$$

(2)

$$\begin{array}{ccc}
\underline{P^*} & & \underline{V(h)} \\
 & \xrightarrow{\;u\;} & \\
 & \xleftarrow{\;c_2 \xleftarrow{\$} \mathbb{Z}_q\;} & \\
 & \xrightarrow{\;z_2\;} & g^{z_2} \stackrel{?}{=} u \cdot h^{c_2}
\end{array}$$

Since we assumed $\epsilon = 1$, then (u, c_1, z_1) and (u, c_2, z_2) are two accepting transcripts. Thus, $g^{z_1} = u \cdot h^{c_1}$ and $g^{z_2} = u \cdot h^{c_2}$

$$\Rightarrow \quad g^{z_1 - z_2} = h^{c_1 - c_2}$$

W.H.P. $c_1 \neq c_2$,

$$g^{\frac{z_1 - z_2}{c_1 - c_2}} = h \quad \Rightarrow \quad x = \frac{z_1 - z_2}{c_1 - c_2} \text{ is the DLog of } h.$$

More formally

$\underline{\mathcal{E}^{P^*}}$

1) Run P^* to get u.
2) Send $c_1 \xleftarrow{\$} \mathbb{Z}_q$, and recieve z_1
3) Rewind P^*, send $c_2 \xleftarrow{\$} \mathbb{Z}_q$, and recieve z_2
4) If $c_1 = c_2$, output fail. O/W output $x = \dfrac{z_1 - z_2}{c_1 - c_2}$.

Analysis

$$\Pr[(h,x)\in R_{PLog} : x \leftarrow \mathcal{E}^{P^*}(h)] = 1 - \frac{1}{q} \geq \Pr[\langle P^*, V\rangle(h)=1] - \frac{1}{q}$$

Thus, $K = \frac{1}{q}$.

We assumed $\Pr[\langle P^*, V\rangle(h)=1] = 1$, but more generally what about $\Pr = \epsilon$?

Rewinding Lemma (BS 19.2)

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

Sigma Protocols (Σ-Protocols)

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

$$\begin{array}{ccc}
\underline{P(x,w)} & & \underline{V(x)} \\
 & \xrightarrow{\quad t \text{ "commitment"} \quad} & \\
 & \xleftarrow{\quad c \text{ "challenge"} \quad} & c \xleftarrow{\$} C \\
 & \xrightarrow{\quad z \text{ "response"} \quad} & \\
 & & \text{Output } 0/1 \text{ deterministically from } (x,t,c,z)
\end{array}$$

Properties:

1) Perfect Completeness

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

$\Rightarrow$ PoK (can you see how?)

3) Special Honest Verifier ZK: $\exists$ efficient $Sim(x, c) \to (t, z)$ s.t. (t, c, z) is an accepting transcript for x.

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

$$\left\{ (t, c, z) : \begin{matrix} c \xleftarrow{\$} C \\ (t, z) \leftarrow Sim(x, c) \end{matrix} \right\} \approx \left\{ View_V \langle P(x, w), V(x) \rangle \right\}$$

☆ In literature, you may see μ-round Sigma Protcols 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 $i \leq \mu$.

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

$$R_{and} := \left\{ \left((x_0, x_1) ; (w_0, w_1) \right) \;\middle|\; (x_0, w_0) \in R_0 \wedge (x_1, w_1) \in R_1 \right\}$$

We can construct a Sigma Protocol (P,V) for R_{and} as follows:

$P((x_0,x_1),(w_0,w_1))$		$V((x_0,x_1))$
Run $P_0(x_0,w_0) \to t_0$ $P_1(x_1,w_1) \to t_1$	$\xrightarrow{(t_0,t_1)}$	
	$\xleftarrow{c}$	$c \xleftarrow{\$} C$
Feed c to P_0, P_1 to obtain (z_0,z_1)	$\xrightarrow{(z_0,z_1)}$	Accept if both $V_0(x_0,(t_0,c,z_0))$ and $V_1(x_1,(t_1,c,z_1))$ accept

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 $\mathcal{E}_0(x_0, (t_0,c,z_0), (t_0,c',z_0')) \to w_0$

$\mathcal{E}_1(x_1, (t_1,c,z_1), (t_1,c',z_1')) \to w_1$

Output (w_0,w_1)

HVZK

$Sim((x_0,x_1),c)$:

$(t_0,z_0) \leftarrow Sim_0(x_0,c)$, $(t_1,z_1) \leftarrow Sim_1(x_1,c)$

output $((t_0,t_1), c, (z_0,z_1))$

$\approx$ Only a sketch: need to argue extractor is correct and simulator distribution is indistinguishable

OR - Proof

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

$P(x_0, x_1, b, w_b)$ $\qquad\qquad\qquad\qquad$ **$V(x_0, x_1)$**

- Compute $c_{\bar{b}} \leftarrow C$
- $(t_{\bar{b}}, z_{\bar{b}}) \leftarrow Sim_{\bar{b}}(x_{\bar{b}}, c_{\bar{b}})$
- Run $P_b(x_b, w_b) \rightarrow t_b$

$$\xrightarrow{\quad (t_0, t_1) \quad}$$

assume C has an XOR $\oplus$ operation

$$\xleftarrow{\quad c \quad} \qquad c \xleftarrow{\$} C$$

- Compute $c_b \leftarrow c \oplus c_{\bar{b}}$
- Feed c_b to $P_b(x_b, w_b)$ get z_b

$$\xrightarrow{\quad (c_0, z_0, z_1) \quad} \qquad c_1 \leftarrow c_0 \oplus c$$

Accept if both $V_0(x_0, (t_0, c_0, z_0))$ and $V_1(x_1, (t_1, c_1, z_1))$ accept

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 $c_1 := c \oplus c_0$ and $c_1' := c' \oplus c_0'$. Notice since $c' \neq c$,

then either $c_0 \neq c_0'$ or $c_1 \neq c_1'$.

$$c_1 \oplus c_1' = (c \oplus c') \oplus (c_0 \oplus c_0')$$

If $c_0 \neq c_0'$:

$\qquad$ Output $(0, \mathcal{E}_0(x_0, (t_0, c_0, z_0), (t_0, c_0', z_0')))$

Else:

$\qquad$ Output $(1, \mathcal{E}_1(x_1, (t_1, c_1, z_1), (t_1, c_1', z_1')))$

HVZK

$Sim((x_0, x_1), c)$

- $c_0 \xleftarrow{\$} C$, $c_1 \leftarrow c \oplus c_0$
- $(t_0, z_0) \leftarrow Sim_0(x_0, c_0)$
- $(t_1, z_1) \leftarrow Sim_1(x_1, c_1)$

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 Prots