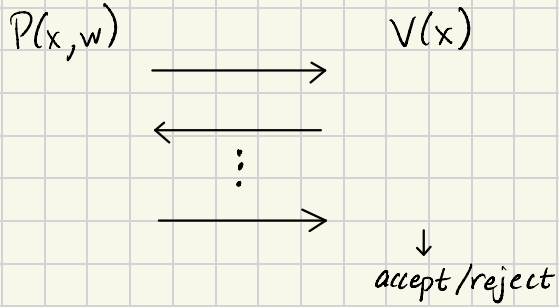
Lecture 10

- Small ZK recap
- Proofs of knowledge
- Schnorr protocol
- Sigma protocols
- Variants (AND/OR)

Recap: zero-knowledge proofs

Each NP language L has associated relation R s.t. $x \in L \Leftrightarrow \exists w$ s.t. $(x,w) \in R$

Ex: L = all graphs with valid 3-colorings
$R = \{(G, \text{valid 3-coloring for } G)\}$



1) Completeness: $\forall x \in L \quad Pr[\langle P(x,w), V(x)\rangle = accept] \geq 1 - \varepsilon$

2) Soundness: $\forall x \notin L \quad \forall P^* \quad Pr[\langle P^*, V(x)\rangle = accept] \leq \varepsilon$

3) Zero-Knowledge: $\forall$ PPT V^*, $\exists$ PPT Sim s.t. $\forall x \in L$

$$\{View_{V^*}[\langle P(x,w), V^*\rangle]\} \approx \{Sim(x)\}$$

↑ "malicious verifier ZK"

- Honest Verifier ZK (HVZK):
  $\exists$ PPT Sim,

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

↑ honest verifier

Proofs of Knowledge

- Soundness (informally): verifier is convinced some x is in an P language L

  x ↓ graph G

  L ↓ 3-colorable graphs

BUT in some cases, we want a stronger guarantee that a prover "knows" a witness to the statement

Soundness: if V accepts, x w.h.p. $\Rightarrow x \in L$

Proof of Knowledge: if V accepts x w.h.p. $\Rightarrow$ P "knows" w

These are not equivalent

Ex: $R_{DL} = \{(g, h \in \mathbb{G}; w) : g^w = h\}$

- let L_h be associated language for R_{DL}
- if $\mathbb{G}$ is cyclic group, then all pairs $(g,h) \in \mathbb{G}^2$ are trivially in L_h $\Rightarrow$ soundness guarantee is trivial, so it does not guarantee knowledge of witness w

How to prove that P^* must know w s.t. $(x,w) \in R$?

Attempt #1: a trivial PoK

$P^*(x,w)$ —— w ——> $V(x)$

$V(x)$: Check $(x,w) \in R$?

- efficient since R is an NP relation
- BUT this cannot be ZK because the simulator would have to efficiently output a witness

Attempt #2

$P^*(x,w)$ $\quad$ $V(x)$

P^* —— m_i ——> V

P^* <—— r_i —— V

- given messages $\{m_i\}$ from P^*, we can compute a satisfying witness...

SAME ISSUE

What if instead we can run the prover multiple times?

Let $\mathcal{E}$ be a PPT algorithm called an extractor that will extract a witness w by cleverly running P^*

↳ $\mathcal{E}$ has black box access to P^*

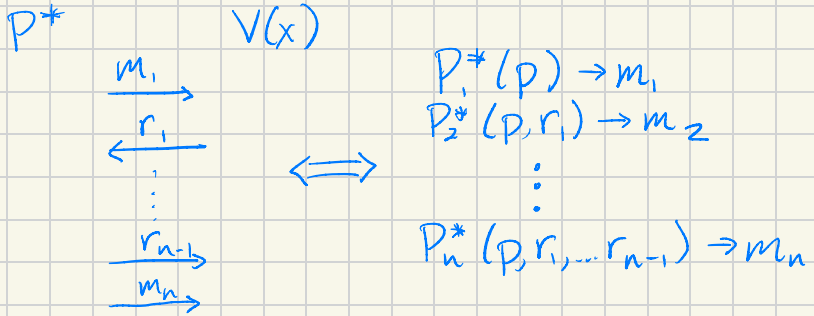
→ can interact with P^*

→ can "rewind" P^* to previous rounds

What is rewinding?

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



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

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

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

Schnorr's Protocol

Fix $\mathbb{G}$ as cyclic group of prime order q, with generator g

Recall $R_{DL} = \{(g, h; w) \mid g^w = h\}$

P receives $w \in \mathbb{Z}_q$, $h = g^w$
V receives h

P wants to convince verifier it knows $w \in \mathbb{Z}_q$ s.t. $g^w = h$

$P(w \in \mathbb{Z}_q, h = g^w \in \mathbb{G})$		$V(h)$
$r \xleftarrow{\$} \mathbb{Z}_q$		
	$u = g^r$ →	
		$c \xleftarrow{\$} \mathbb{Z}_q$ ↑ challenge
	← c	
	$z = r + cw$ →	
		check $g^z = u \cdot h^c$

Claim: Schnorr's Protocol is a ZKPoK for R_{DL}

zero-knowledge proof of knowledge

Proof

1. Completeness: $u h^c = g^r \cdot (g^w)^c = g^z$

2. Honest Verifier Zero Knowledge (HVZK)

We construct a simulator Sim

Sim(h):
$z \xleftarrow{\$} \mathbb{Z}_q$, $c \xleftarrow{\$} \mathbb{Z}_q$
$u \leftarrow g^z / h^c$

The basic idea is Sim runs the protocol "in reverse," which allows it to forge transcript w/o knowing w

Output (u, c, z)

$$\Pr[\text{Sim}(h) = (u,c,z)] = \begin{cases} \frac{1}{q} \cdot \frac{1}{q} & \text{if } u = g^z/h^c \\ \uparrow \text{sample } z \quad \uparrow \text{sample } c & \\ 0 & \text{otherwise} \end{cases}$$

$$\Pr\left[[\text{View}_V\langle P(h,w), V(h)\rangle] = (u,c,z)\right] = \begin{cases} \frac{1}{q} \cdot \frac{1}{q} & \text{if } g^z = u \cdot h^c \\ \uparrow \text{sample } r \quad \uparrow \text{sample } c & \\ 0 & \text{otherwise} \end{cases}$$

$\Rightarrow$ we have HVZK!

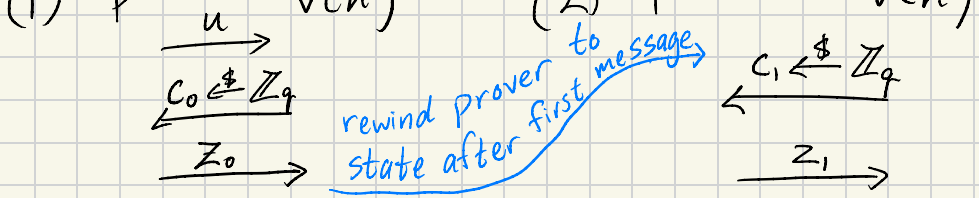
Why not malicious? Malicious verifier doesn't have to choose c randomly

Solution!
- Verifier commits to c before seeing u

3. Proof of Knowledge

Suppose P^* is a (possibly malicious) prover that convinces an honest verifier w.p. ε (for sake of simplicity, assume $\varepsilon = 1$; general case in Boneh-Shoup 19.1)

Intuition: Rewind the prover to operate on different challenges



Since we assumed $\varepsilon = 1$, (u, c_0, z_0) and (u, c_1, z_1) are two accepting transcripts, so

$$g^{z_0} = u \cdot h^{c_0} \quad \text{and} \quad g^{z_1} = u \cdot h^{c_1}$$

$$\Rightarrow g^{z_0 - z_1} = h^{c_0 - c_1}$$

W.h.p., $c_0 \neq c_1$,

$$g^{\frac{z_0 - z_1}{c_0 - c_1}} = h \Rightarrow \frac{z_0 - z_1}{c_0 - c_1} \text{ is DLog of } h$$

More formally:

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

1. Run P^* to get u
2. Send random challenge $c_0 \xleftarrow{\$} \mathbb{Z}_q$; receive z_0
3. Rewind P^*, send $c_1 \xleftarrow{\$} \mathbb{Z}_q$; receive z_1
4. If $c_0 = c_1$, output fail. O.w. output $x = \frac{z_0 - z_1}{c_0 - c_1}$

Analysis:

- $\mathcal{E}^{P^*}$ fails when $c_0 = c_1$ $\Rightarrow$ occurs w.p. $1/q$

$$\Pr[(h,w) \in R_{DL} : w \leftarrow \mathcal{E}^{P^*}(h)] = 1 - 1/q$$

$$\geq \Pr[\langle P^*, V \rangle(h) = 1] - 1/q$$

$\Rightarrow$ knowledge error $\kappa = 1/q$

Digest

Q: It might seem PoK and ZK are contradictory:
if $\mathcal{E}$ can learn w from P^*, why can't V?

A. $\mathcal{E}$ and V interact with P^* in very different ways.
$\mathcal{E}$ has <u>much</u> more power than V
↳ V must interact with P^* in "live protocol"
↳ $\mathcal{E}$ can "rewind" P^*

Sigma Protocols (Σ-Protocols)

- a more general view of Schnorr Protocol

$P((x,w) \in R)$ $\qquad$ $V(x)$

t ("commitment") →

← c ("challenge")

z ("response") →

$c \xleftarrow{\$} C$

challenge is chosen uniformly at random

outputs accept/reject as a deterministic function of (x,t,c,z)

Properties

1. Perfect completeness
2. Special Soundness: $\exists$ efficient extractor $\mathcal{E}$ that, given two accepting transcripts $(t,c,z), (t,c',z')$ s.t. $c \neq c'$ outputs w s.t. $(x,w) \in R$
   ↳ can show special soundness $\Rightarrow$ PoK with $\kappa = 1/q$
3. Special Honest Verifier Zero-Knowledge: $\exists$ efficient simulator Sim that takes (x,c) as input s.t.
   a) $Sim(x,c) \to (t,z)$ s.t. (t,c,z) is an accepting transcript for x
   b) $\forall (x,w) \in R$

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

$\equiv$: identically distributed

Why do we care about Sigma Protocols?

- efficient ZK proofs for many interesting languages
- can build efficient identification protocols and signature schemes

Composition of Σ-Protocols

Let (P_0, V_0) be a Σ-protocol for R_0
Let (P_1, V_1) be a Σ-protocol for R_1

We want to construct Σ-protocol for R_{AND}

$$R_{AND} = \left\{ (x_0, x_1), (w_0, w_1) : \begin{matrix} (x_0, w_0) \in R_0 \\ (x_1, w_1) \in R_1 \end{matrix} \right\}$$

$\Rightarrow$ can just run Σ-protocols for R_0 and R_1 in parallel (can even use same challenge)

$P((x_0, x_1), (w_0, w_1))$ | $V((x_0, x_1))$

Run $P_0(x_0, w_0) \to t_0$
$P_1(x_1, w_1) \to t_1$

(t_0, t_1) →

$c \xleftarrow{\$} C$

← c

Feed c to P_0, P_1 to obtain (z_0, z_1)

(z_0, z_1) →

Accept if both $V(t_0, c, z_0)$ and $V(t_1, c, z_1)$ output accept

Claim: The above is Σ-protocol for R_{AND}

Pf. Intuition:

- Completeness obvious
- Special Soundness: get 2 accepting transcripts
  $((t_0, t_1), c, (z_0, z_1))$, $((t_0, t_1), c', (z_0', z_1'))$
  Run extractors $\mathcal{E}_0((t_0, c, z_0), (t_0, c', z_0')) \to w_0$
  $\mathcal{E}_1((t_1, c, z_1), (t, c', z_1')) \to w_1$
  Output (w_0, w_1)

- SHVZK

$Sim_{AND}((x_0, x_1), c)$:

$(t_0, z_0) \leftarrow Sim_0(x_0, c)$

$(t_1, z_1) \leftarrow Sim_1(x_1, c)$

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

What about Σ-protocol for R_{OR}

$$R_{OR} = \{(x_0, x_1); (b, w_b) : (x_b, w) \in R_b\}$$

High Level Ideas:

- Verifier sends challenge $c \in \{0,1\}^n$
- Prover chooses c_0, c_1 s.t. $c_0 \oplus c_1 = c$ and creates one real proof and one simulated proof

$P((x_0, x_1); (b, w))$ | $V((x_0, x_1))$

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

(t_0, t_1) →

$c \xleftarrow{\$} C$

← c

Compute $c_b = c \oplus c_{\bar{b}}$

Feed c_b to P_b to obtain z_b

(c_0, z_0, z_1) →

$c_1 \leftarrow c \oplus c_0$

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

Claim: The above is Σ-protocol for R_{OR}

Pf. Sketch

1. Completeness: obvious

2. Special Soundness:

Given $((t_0, t_1), c, (c_0, z_0, z_1)), ((t_0, t_1), c', (c_0', z_0', z_1'))$,
let $c_1 = c \oplus c_0$ and $c_1' = c' \oplus c_0'$
Since $c \neq c'$, either $c_0 \neq c_0'$ or $c_1 \neq c_1'$
If $c_0 \neq c_0'$, then:
Run $\mathcal{E}_0((t_0, c_0, z_0), (t_0, c_0', z_0')) \to w_0$
Output $(0, w_0)$
Else:
Run $\mathcal{E}_1((t_1, c_1, z_1), (t_1, c_1', z_1')) \to w_1$
Output $(1, w_1)$

3. SHVZK
$\text{Sim}_{OR}((x_0, x_1), c)$
- $c_0 \xleftarrow{\$} C$, $c_1 \leftarrow c \oplus c_0$
- $(t_0, z_0) \leftarrow \text{Sim}_0(x_0, c_0)$
- $(t_1, z_1) \leftarrow \text{Sim}_1(x_1, c_1)$

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