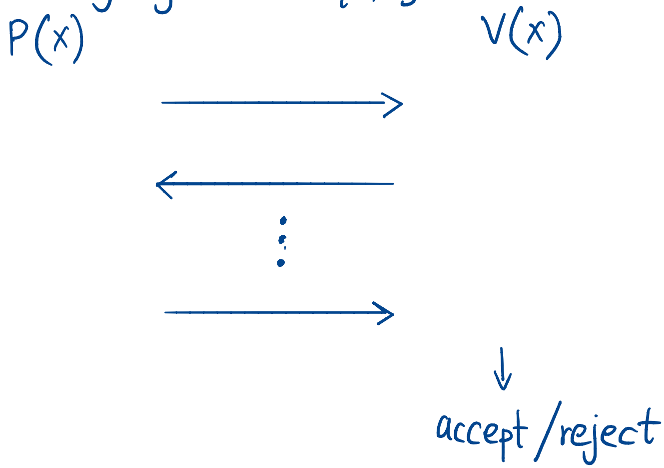
Recap: zero-knowledge proofs

Language $L \subset \{0,1\}^*$



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

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

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

$$\left\{ \text{View}_{V^*}[\langle P, V^*\rangle(x)] \right\} \approx_c \left\{ \text{Sim}(x) \right\}$$

__Theorem__: $\exists \text{OWF} \Rightarrow \text{NP} \subset \text{ZK}$

__Proof:__ We saw a ZK proof for HAMCYCLE

Applications of ZK:

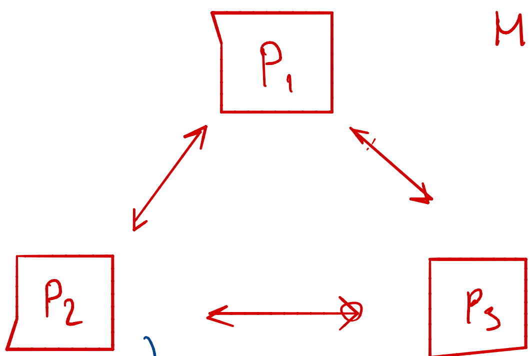
(1) Identification protocols (& signatures) ↪ next lecture

ex: prove to server you know "password" without revealing it.

(2) Enforce honest behaviour in protocols

We will talk about MPC in 2 weeks

schematically:



wants to prove that messages sent follow protocol without disclosing secrets

Proof of Knowledge:

Soundness property assures verifier that some NP statement is true.

Sometimes we want a stronger guarantee: that the prover "knows" a witness to the statement.

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

Example: L - all hamiltonian graphs

R - $\{(G, \text{ham. path in } G)\}$

Soundness - if V accept x w.h.p $\Rightarrow x \in L$

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

These are not equivalent: example:

$$R = \{(N, p) : p \mid N, p \neq 1, N\}$$

How can we prove that P must know w?

(- Cannot look for w in the code of P)

- Can EXTRACT w by (cleverly) running P

<u>Defn:</u> $\langle P, V \rangle$ is a PoK for R if $\exists$ PPT E (called an "extractor") s.t. $\forall x \; \forall P^*$

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

E can run P^*

knowledge error

Schnorr's Protocol

Fix G cyclic group of order q, g generator.

P is given $x \in \mathbb{Z}_q$, $h = g^x$. V is given h. P wants to convince verifier it knows $x \in \mathbb{Z}_q$ s.t. $g^x = h$.

$$L = \{ h \in G \mid \exists x \in \mathbb{Z}_q : h = g^x \}$$

$$R = \{ (h, x) \mid h \in G, x \in \mathbb{Z}_q \text{ s.t } h = g^x \}$$

Since every group element is in L, proving that $h \in L$ is trivial. That's why it only makes sense to talk about a proof of knowledge for R.

$P(x \in \mathbb{Z}_q, h = g^x \in G)$		$V(h \in G)$
$r \xleftarrow{\$} \mathbb{Z}_q$		
	$\xrightarrow{u = g^r}$	
	$\xleftarrow{c}$	$c \xleftarrow{\$} \mathbb{Z}_q$
	$\xrightarrow{z = r + cx}$	
		Check
		$g^z = u \cdot h^c$

Claim: Schnorr's protocol is a ZKPoK of DLOG.

zero knowledge proof of knowledge

Proof: (1) Completeness: $u \cdot h^c = g^r \cdot (g^x)^c = g^{r+xc} = g^z$.

(2) Honest-verifier zero knowledge:

We construct a simulator S.

$\underline{S(h):}$

$z \xleftarrow{\$} \mathbb{Z}_q$, $c \xleftarrow{\$} \mathbb{Z}_q$

$u \leftarrow g^z / h^c$

output (u, c, z)

The basic idea: S runs the protocol "in reverse", which allows it to forge transcript w/o knowing x.

Claim: $\{ S(h) \} \approx \{ \mathrm{View}_V \langle P, V \rangle (h) \}$

honest verifier

Proof of Claim:

$$\Pr[S(h) = (u, c, z)] = \Pr[\mathrm{View}_V \langle P, V \rangle (h) = (u, c, z)] = \begin{cases} 1/q^2 & : u = g^z / h^c \\ 0 & : \text{o.w.} \end{cases}$$

Why is this only HV ZK? a malicious verifier doesn't have to choose at random.

We will discuss malicious-verifier ZK later.

(3) Proof of Knowledge

Suppose P^* is a (possibly malicious) prover that convinces honest verifier w.p. ε.

And for the sake of simplicity, suppose $\varepsilon = 1$

(See Boneh-Shoup 19.1 for general case.)

Let E be the following extractor:

1. Run P^* to obtain initial message u
2. Send a random challenge $c_1 \xleftarrow{\$} \mathbb{Z}_q$, get back z_1
3. Rewind the prover to its state after the 1st message.
4. Send it another random challenge $c_2 \xleftarrow{\$} \mathbb{Z}_q$, get z_2
5. Output $x = \dfrac{z_2 - z_1}{c_2 - c_1} \in \mathbb{Z}_q$

Analysis: since P^* succeeds w.p. 1, we know that

$$g^{z_1} = u \cdot h^{c_1} \qquad g^{z_2} = u \cdot h^{c_2}$$

Therefore: $$g^{z_1} / h^{c_1} = g^{z_2} / h^{c_2} \Rightarrow g^{z_1 - xc_1} = g^{z_2 - xc_2} \Rightarrow x = \frac{z_1 - z_2}{c_1 - c_2}$$

Digest:

It might seem that PoK and ZK are perhaps contradictory: if E can learn w from P why can't the verifier V?

Answer: E and V interact with P^* in very different ways. Specifically, E has much more power than V:

V must interact with P^* in "live protocol"

E can rewind P^*.

HVZK $\Rightarrow$ ZK

Problem: V can choose c not uniformly random.

Strawman: Run P to get first msg u.

Feed u to V to get c

Run Sim to get (u', c, z) (probably $u' \neq u$)

Problem: c can depend on u. So (u', c, z) doesn't look like real transcript.

Solution: V first commits to c, then V can't change c depending on u.

Sigma protocols

A more general view of Schnorr's protocol:

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

$$\xrightarrow{t \text{ ("commitment")}}$$

$$\xleftarrow{c \text{ ("challenge")}}$$

$c \xleftarrow{\$} C$

challenge is chosen uniformly at random

$$\xrightarrow{z \text{ ("response")}}$$

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

Requirements

(1) Completeness (perfect)

(2) Special soundness: there exist an eff. extractor 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 Sp. Soundness $\Rightarrow$ PoK with $\kappa = 1/|C|$.

(3) Special Honest Verifier ZK:

There exists an efficient Sim that takes as input (x, c) and s.t:

(1) $\mathrm{Sim}(x, c)$ outputs t, z s.t (t, c, z) is an accepting transcript for x.

(2) For all $(x, w) \in R$

$$\left\{ (t, c, z) : \begin{array}{l} c \xleftarrow{\$} C \\ (t, z) \leftarrow \mathrm{Sim}(x, c) \end{array} \right\} \equiv \left\{ \mathrm{View}_V \left(P(x, w) \leftrightarrow V(x) \right) \right\}$$

identically distributed

Why Sigma Protocols are interesting?

- Efficient ZK proofs for many interesting languages. (about commitments, encryptions, ...)

- Can build efficient identification protocols & <u>signature schemes</u> $\longrightarrow$ Next lecture

Composition of Σ-protocols

Given Σ-protocol for $R = \{(x,w)\}$

want to construct Σ-protocols for

- Proving AND of statements

$$R_{AND} = \left\{ ((x_0, x_1), (w_0, w_1)) : \begin{array}{l} (x_0, w_0) \in R \\ (x_1, w_1) \in R \end{array} \right\}$$

$\Rightarrow$ just run protocols in parallel (can even use same challenge)

can also do AND/OR of diff. relations

- Proving OR of statements

$$R_{OR} = \left\{ ((x_0, x_1), (b, w)) : \underset{b \in \{0,1\}}{(x_b, w) \in R} \right\}$$

This is much trickier. Basic idea:
Verifier sends challenge $c \in \{0,1\}^n$
Prover can choose c_0, c_1 s.t $c_0 \oplus c_1 = c$
and then create one real proof and one simulated proof. Verifier doesn't know which is which.

$P_{OR}((x_0, x_1), (b, \omega))$ $\qquad\qquad$ $V_{OR}(x_0, x_1)$

Suppose $b=0$ (i.e. prover knows witness to R)

$c_1 \xleftarrow{\$} C$

Run Sim_1 for R_1 to get

(t_1, c_1, z_1) valid transcript

$t_0 \leftarrow P(x_0, \omega)$

$$\xrightarrow{t_0,\, t_1}$$

$$\xleftarrow{c \xleftarrow{\$} C}$$

$c_0 \leftarrow c \oplus c_1$

Send c_0 to P

to get response z_0

$$\xrightarrow{c_0,\, z_0,\, z_1}$$

check

(x_0, t_0, c_0, z_0)

$(x_1, t_1, c \oplus c_0, z_1)$

(1) Completeness - immediate

(2) SHVZK - we construct Sim_{OR}:

choose $c \xleftarrow{\$} \mathbb{Z}_q$, $c_0 \xleftarrow{\$} \mathbb{Z}_q$, set $c_1 \leftarrow c_0 \oplus c$

We can now use Sim for R to compute:

$$(t_0, z_0) \leftarrow \mathrm{Sim}(x_0, c_0)$$

$$(t_1, z_1) \leftarrow \mathrm{Sim}(x_1, c_1)$$

output $\left((t_0, t_1), (z_0, z_1)\right)$

(3) Special Soundness

Given (x_0, x_1) & two transcripts

$\left((t_0, t_1), c, (c_0, z_0, z_1)\right)$ & $\left((t_0, t_1), c', (c_0', z_0', z_1')\right)$

s.t. $c \neq c'$

Define $c_1 = c \oplus c_0$ $\quad c_1' = c' \oplus c_0'$

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

(since $c \neq c'$)

Suppose w.l.o.g. $c_0' \neq c_0$

Then can use extractor for R to extract w_0 from

$$w_0 \leftarrow \mathrm{Ext}\Big(x_0, (t_0, c_0, z_0), (t_0, c_0', z_0')\Big)$$

and then output witness

$(0, w_0)$ for R_{OR}.