Recap: zero-knowledge proofs

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

P(x) $\overset{\rightarrow}{\longrightarrow}$ V(x)

$\overset{\leftarrow}{\rightarrow}$

$\overset{\downarrow}{\rightarrow}$ accept/reject

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

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

3. Zero Knowledge: $\forall \text{PPT} \forall \text{PPT Sim} \quad \forall x \in L$

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

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

(2) Enforce honest behaviour in protocols

schematically:

\[ P_1 \rightarrow P_2 \leftarrow P_3 \]

\( \uparrow \)

wants to prove that messages sent follow protocol \( \rightarrow \) without disclosing secrets

\( \Rightarrow \)

We will talk about MPC in 2 weeks.
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. $x \in L \iff \exists w \text{ s.t. } (x, w) \in R$

Example: $L$ - all hamiltonian graphs
$R = \{(G, \text{ ham. path in } G)\}$

Soundness - if $V$ accepts $x$ 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:
$R = \{ (N, p) : p | N, p \neq 1, N \}$
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

---

**Defn:** \( \langle P, V \rangle \) is a Pok for R if \( \exists \text{PPT } E \) (called an “extractor”) s.t. \( \forall x \forall P^* \)

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

\( E \) can run \( P^* \)

\( \kappa \) 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 $R$.

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

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

$$u = g^r$$

$$c \leftarrow \mathbb{Z}_q$$

$$z = r + cx$$

Check

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

$V(heG)$
Claim: Schnorr's protocol is a ZKPoK of DLOG.

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

(2) Honest-verifier zero knowledge:
We construct a simulator \( S \).

\[
S(h):
\begin{align*}
& z \in Z_q, \ c \in Z_q \\
& u \leftarrow g^2/c \\
& \text{output } (u, c, z)
\end{align*}
\]

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

\[
\frac{1}{q^2} : u = g^2/c
\]

\[
\Pr\left[S(h) = (u, c, z)\right] = \Pr\left[\text{View}_v \langle P, V \rangle(h) \ni (u, c, z)\right] = \begin{cases} 
0 & : \text{w. o.} 
\end{cases}
\]

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. $\varepsilon$. 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 $c_1 \in \mathbb{Z}_q$, get back $z_1$
3. Rewind the prover to its state after the 1st message.
4. Send it another random challenge $c_2 \in \mathbb{Z}_q$, get $z_2$
5. Output $x = \frac{z_2 - z_1}{c_2 - c_1} \in \mathbb{Z}_q$

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

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

Therefore:

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

\[ P((x, w) \in R) \quad V(x) \]

\[ \begin{align*}
  t \quad & \text{("commitment")} \\
  c \quad & \text{("challenge")} \\
  z \quad & \text{("response")} \\
\end{align*} \]

\[ c \leftarrow \$ \quad C \]

Challenge is chosen uniformly at random.

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. \( (x, w) \in R \)

\( \Rightarrow \) can show Sp. Soundness \( \Rightarrow \) PoK with \( R = \frac{1}{\sqrt{C}} \).
(3) Special Honest Verifier ZK:

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

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

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

\[
\{ (t, c, z) : \exists c \in C \text{ s.t. } (t, z) \leftarrow \text{Sim}(x, c) \} \equiv \{ \text{View}_V(P(x, w) \leftrightarrow V(x)) \} \]

identically distributed

Why Sigma Protocols are interesting?

- Efficient ZK proofs for many interesting languages. (about commitments, encryptions, ...)
- Can build efficient identification protocols & signature schemes. → Next lecture
Composition of $\Sigma$-protocols

Given $\Sigma$-protocol for $R = \{(x, w)\}$ want to construct $\Sigma$-protocols for.

- Proving AND of statements

$$R_{\text{AND}} = \left\{ \left\langle (x_0, x_1), (w_0, w_1) \right\rangle : (x_0, w_0) \in R \land (x_1, w_1) \in R \right\}$$

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

- Proving OR of statements

$$R_{\text{OR}} = \left\{ \left\langle (x_0, x_1), (b, w) \right\rangle : (x_b, w) \in R \right\}$$

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

\[
P_{OR}(x_0, x_1), (b, \omega) \quad \frac{\text{Suppose } b=0 \text{ (i.e. prover knows witness to } R)}{V_{OR}(x_0, x_1)}
\]

\[C_1 \leftarrow C\]

Run \( \text{Sim}_1 \) for \( R_1 \) to get

\[(t_1, c_1, z_1) \text{ valid transcript}\]

\[t_0 \leftarrow P(x_0, \omega)\]

\[\frac{t_0, t_1}{t_0, t_1} \xrightarrow{} c \leftarrow C\]

\[c_0 \leftarrow C \oplus C_1\]

Send \( c_0 \) to \( P \)

to get response \( z_0 \)

\[c_0, z_0, z_1 \xrightarrow{} \text{check } (x_0, t_0, c_0, z_0)\]

\[(x_1, t_1, C \oplus C_0, z_1)\]
(1) Completeness - immediate

(2) SHVZK - we construct Sim or: choose \( c \leftarrow \mathbb{Z}_q \), \( c_0 \leftarrow \mathbb{Z}_q \), set \( c_1 \leftarrow c_0 \oplus c \)

We can now use Sim for R to compute:

\[
(t_0, z_0) \leftarrow \text{Sim}(x_0, c_0) \\
(t_1, z_1) \leftarrow \text{Sim}(x_1, c_1)
\]

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

(3) Special Soundness

Given \((x_0, x_1)\) & two transcripts

\[
( (t_0, t_1), c, (c_0, z_0, z_1) ) \quad \& \quad ( (t_0, t_1), c_1, (c_0', z_0', z_1') )
\]

s.t. \( c \neq c_1 \)
Define $C_1 = C \oplus C_0$, $C'_1 = C' \oplus C_0$

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

$w_0 \leftarrow \text{Ext}(x_0, (t_0, C_0, z_0), (t_0, C'_0, z'_0))$

and then output witness

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