In many cases, we want a stronger property: the prover actually "knows" why a statement is true (e.g., it knows a "witness")

For instance, consider the following language:

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

group of order p — generator of $\mathbb{G}$

Note: this definition of $\mathcal{L}$ implicitly defines an NP relation R:
$R(h, x) = 1 \iff h = g^x \in \mathbb{G}$

In this case, all statements in $\mathbb{G}$ are true (i.e., contained in $\mathcal{L}$), but we can still consider a notion of proving knowledge of the discrete log of an element $h \in \mathbb{G}$ — conceptually stronger property than proof of membership

Philosophical question: What does it mean to "know" something?

If a prover is able to convince an honest verifier that it "knows" something, then it should be possible to extract that quantity from the prover.

Definition. An interactive proof system (P, V) is a proof of knowledge for an NP relation R if there exists an efficient extractor $\mathcal{E}$ such that for any x and any prover P^*

proof of knowledge is parameterized by a specific relation R (as opposed to the language $\mathcal{L}$)

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

more generally, could be polynomially smaller

knowledge error

Trivial proof of knowledge: prover sends witness in the clear to the verifier
↳ In most applications, we additionally require zero-knowledge

Note: knowledge is a strictly stronger property than soundness
↳ if protocol has knowledge error ε $\Rightarrow$ it also has soundness error ε (i.e. a dishonest prover convinces an honest verifier of a false statement with probability at most ε)

Proving knowledge of discrete log (Schnorr's protocol)

assume $g, h \in \mathbb{G}$
where $\mathbb{G}$ has prime order q

Suppose prover wants to prove it knows x such that $h = g^x$ (i.e. prover demonstrates knowledge of discrete log of h base g)

prover		verifier
$r \xleftarrow{R} \mathbb{Z}_p$ $u \leftarrow g^r$	$\xrightarrow{\quad u \quad}$	
	$\xleftarrow{\quad c \quad}$	$c \xleftarrow{R} \mathbb{Z}_p$
$z \leftarrow r + cx$	$\xrightarrow{\quad z \quad}$	

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

Completeness: if $z = r + cx$, then

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

zero knowledge only required to hold against an honest verifier (e.g., view of the honest verifier can be simulated)

Honest-Verifier Zero-Knowledge: build a simulator as follows (familiar strategy: run the protocol in "reverse"):

on input (g, h):

1. Sample $z \xleftarrow{R} \mathbb{Z}_p$
2. Sample $c \xleftarrow{R} \mathbb{Z}_p$
3. set $u = g^z / h^c$ and output (u, c, z)

uniformly random challenge

uniformly random group element since z is uniformly random

chosen so that $g^z = u \cdot h^c$ (relation satisfied by a valid proof)

Simulated transcript is identically distributed as the real transcript with an honest verifier

What goes wrong if the challenge is not sampled uniformly at random (i.e., if the verifier is dishonest)

Above simulation no longer works (since we cannot sample z first)

↳ To get general zero knowledge, we require that the verifier first commit to its challenge (using a statistically hiding commitment)

Knowledge: Suppose P^* is (possibly malicious) prover that convinces honest verifier with probability 1. We construct an extractor as follows:

for simplicity, we assume P^* succeeds with probability 1

1. Run the prover P^* to obtain an initial message u.
2. Send a challenge $c_1 \xleftarrow{R} \mathbb{Z}_p$ to P^*. The prover replies with a response z_1.
3. "Rewind" the prover P^* so its internal state is the same as it was at the end of Step 1. Then, send another challenge $c_2 \xleftarrow{R} \mathbb{Z}_p$ to P^*. Let z_2 be the response of P^*.
4. Compute and output $x = (z_1 - z_2)(c_1 - c_2)^{-1} \in \mathbb{Z}_p$.

Since P^* succeeds with probability 1 and the extractor perfectly simulates the honest verifier's behavior, with probability 1, both (u, c_1, z_1) and (u, c_2, z_2) are both accepting transcripts. This means that

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

$$\Rightarrow \frac{g^{z_1}}{h^{c_1}} = \frac{g^{z_2}}{h^{c_2}} \Rightarrow g^{z_1 + c_2 x} = g^{z_2 + c_1 x}$$

$$\Rightarrow x = (z_1 - z_2)(c_1 - c_2)^{-1} \in \mathbb{Z}_p$$

with overwhelming probability, $c_1 \neq c_2$

Thus, extractor succeeds with overwhelming probability.

(Boneh-Shoup, Lemma 19.2)

If P^* succeeds with probability ε, then need to rely on "Rewinding Lemma" to argue that extractor obtains two accepting transcripts with probability at least $\varepsilon^2 - 1/p$.

The ability to extract a witness from any two accepting transcripts is very useful

↳ called special soundness (for 3-message protocols)

given (u, t_1, z_1) and (u, t_2, z_2) ⟹ can extract the witness

initial message — challenge — response

[same initial message, different challenges]

3-message protocols that satisfy completeness, special soundness, and HVZK are called Σ-protocols
↳ Σ-protocols are useful for building signatures and identification protocols

How can a prover both <u>prove knowledge</u> and yet be zero-knowledge at the same time?
↳ Extractor operates by "rewinding" the prover (if the prover has good success probability, it can answer most challenges correctly.
↳ But in the real (actual) protocol, verifier <u>cannot</u> rewind (i.e., verifier only sees prover on fresh protocol executions), which can provide zero-knowledge.

Many extensions of Schnorr's protocol to prove relations in the exponent.

For example, suppose we want to prove that (g, h, u, v) is a DDH tuple (i.e., $\exists x \in \mathbb{Z}_p : h = g^x$ and $v = u^x$)
<u>Chaum-Pedersen</u>:

<u>prover</u>	<u>verifier</u>

$\xrightarrow{\quad g^r \quad u^r \quad}$

$\xleftarrow{\quad t \quad}$

$\xrightarrow{\quad r + xt \quad}$

check that
$$g^{r+xt} \stackrel{?}{=} (g^r) h^t$$
$$u^{r+xt} \stackrel{?}{=} (u^r) v^t$$

<u>Completeness</u>: Follows by construction.
<u>Special soundness</u>: Suppose we have accepting transcripts
$((\alpha, \beta), t_1, z_1)$ and $((\alpha, \beta), t_2, z_2)$
Then
$$\frac{g^{z_1}}{g^{z_2}} = \frac{\alpha h^{t_1}}{\alpha h^{t_2}} \Rightarrow z_1 - z_2 = x(t_1 - t_2) \quad (\text{mod } p)$$
$$\Rightarrow x = (z_1 - z_2)(t_1 - t_2)^{-1}$$

Similar calculation shows that $v = u^x$.

<u>HVZK</u>: Construct simulator as follows:
1. Sample $z \xleftarrow{R} \mathbb{Z}_p$
2. Sample $t \xleftarrow{R} \mathbb{Z}_p$.
3. Output $(g^z / h^t, v^z / u^t, t, z)$

Since z is uniform and $h = g^x$, $v = u^x$, distribution of $(g^z/h^t, u^z/v^t)$ is identical to (g^r, u^r).
Challenge t is identically distributed, which uniquely determines z.