Zero Knowledge - Session 4

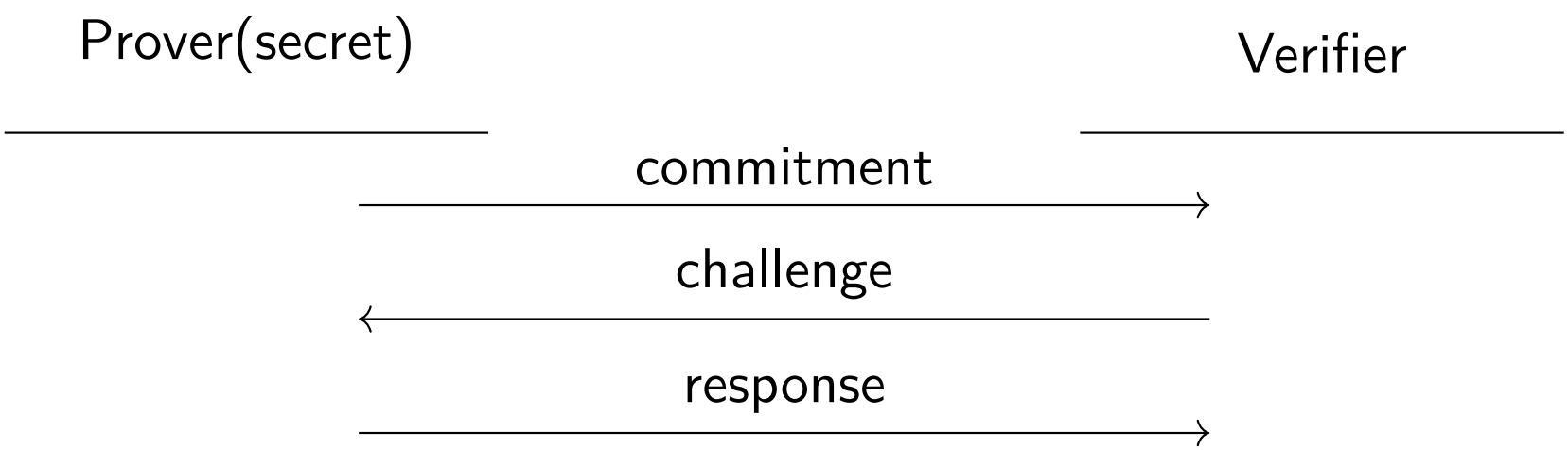
Elahe Sadeghi

October 2020

Σ-protocols

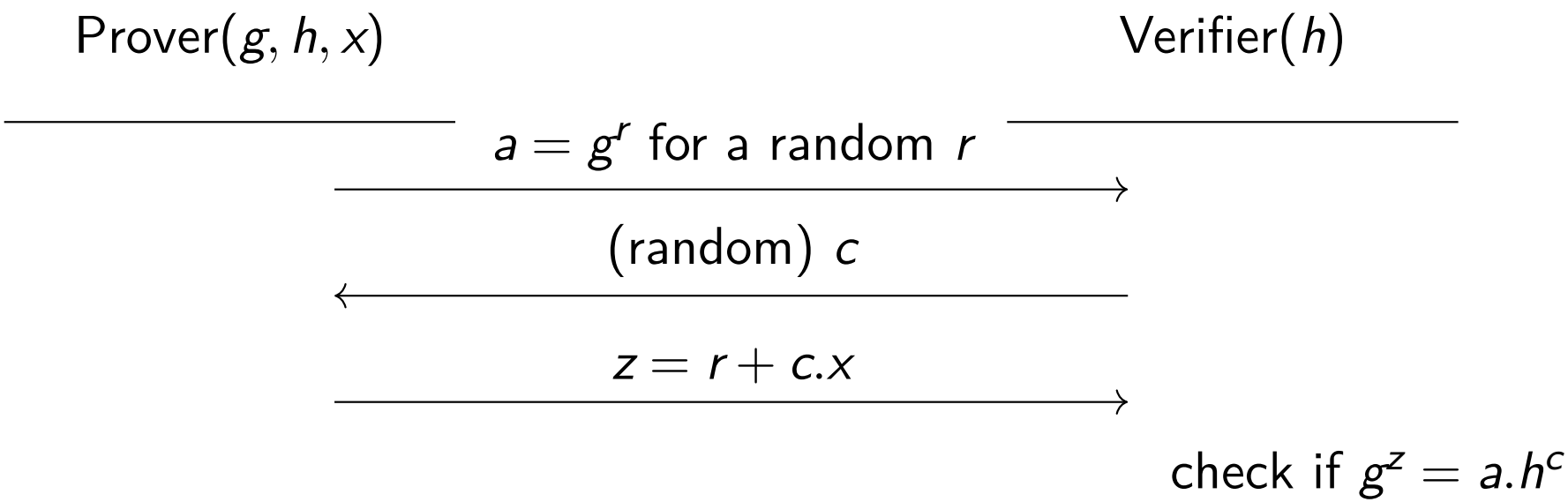
So far, we saw every language in $\mathcal{NP}$ has a zero knowledge proof.
A general view of this is Σ-protocols. Σ-protocols have a structure of "commitment-challenge-response".

- Such construction is as follows:



Σ-protocols

- **Schnorr's protocol**: (an example of Σ-protocols) Assume a cyclic group G with generator g. Prover wants to convince the verifier that she knows a secret x such that $g^x = h$. (which h is public)



Note **Discrete Log assumption**: Given $h = g^x$, computing x is hard!

* **And-proof**: proof of knowledge of two statements.
In Schnorr's protocol, prover knows x_1,x_2 such that $g_1^{x_1} = h_1$ and $g_2^{x_2} = h_2$ for fixed g_1,g_2.

Prover — Verifier

$(a_1 = g_1^r, a_2 = g_2^r)$ for a random r →

← c

$z = r + c.x$ →

check if $\begin{cases} g_1^z = a_1.h_1^c \\ g_2^z = a_2.h_2^c \end{cases}$

Σ-protocols

* **Or-proof**: proving or of two statements without revealing which one is true.

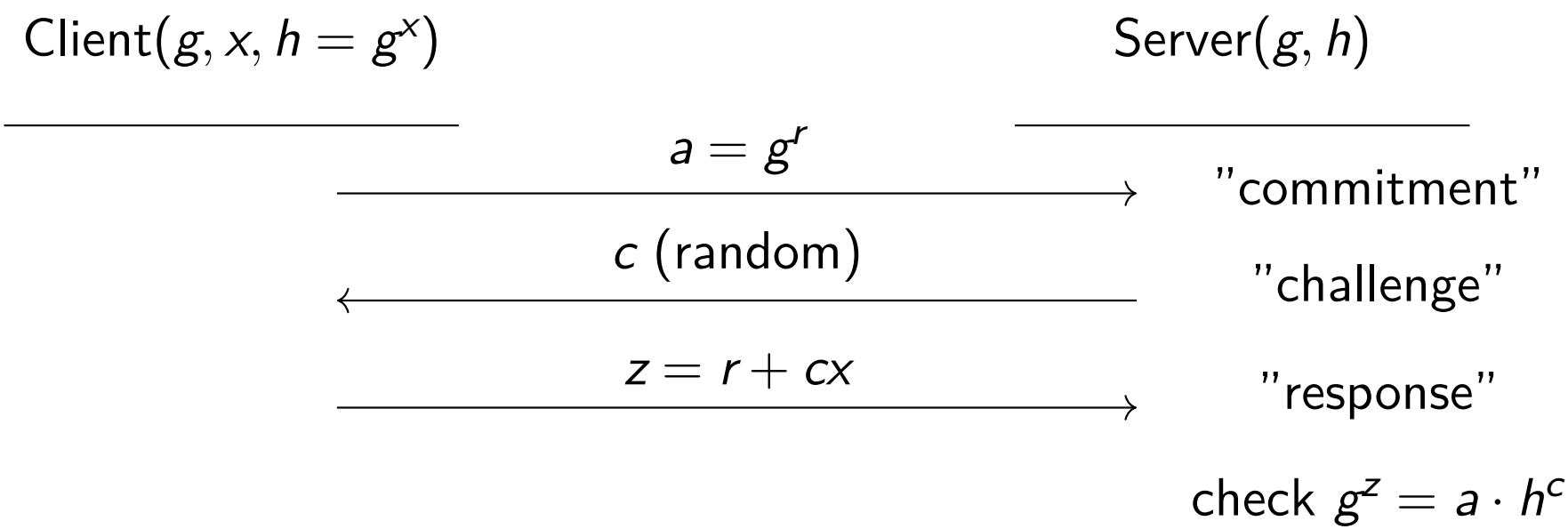
  idea Prover will simulate the transcript it doesn't know.

  In Schnorr's protocol, assume prover knows x_1:

  1. prover runs Schnorr's simulator on input (g, h_2) and obtains transcript (a_2, c_2, z_2).
  2. prover samples random r_1 and computes $a_1 = g^{r_1}$.
  3. prover sends (a_1, a_2) and receives challenge c.
  4. prover computes $c_1 = c - c_2$ and $z_1 = r_1 + c_1.x_1$. Then sends (c_1, c_2, z_1, z_2).
  5. verifier checks if $c_1 + c_2 = c$ and $\begin{cases} g_1^z = a_1.h_1^c \\ g_2^z = a_2.h_2^c \end{cases}$.

Σ-protocols

Example: Schnorr's protocol - an identification protocol



Σ-protocols

-**Completeness**: Trivial!
-**(Honest-Verifier) Zero-Knowledge**: Construct the simulator as follows:

1. sample z random.
2. sample c random.
3. set $a = \frac{g^z}{h^c}$ and output (a, c, z).

-**Soundness**: If clinet does not know x, the best he can do is prepare a such that he can response to server's challenge properly. So the soundness error will be the probability of client guessing c correctly, which is $\frac{1}{|c|}$.

Σ-protocols

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

1. Run the prover P^* to obtain an initial message a.
2. Send a challenge c_1 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 to P^*. Let z_2 be the response of P^*.
4. Compute and output $x = (z_1 - z_2)(c_1 - c_2)^{-1}$.

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

$$g^{z_1} = a \cdot h^{c_1} \quad \textit{and} \quad g^{z_1} = a \cdot h^{c_1} \Rightarrow \frac{g^{z_1}}{h^{c_1}} = \frac{g^{z_2}}{h^{c_2}} \Rightarrow g^{z_1 + c_2 x} = g^{z_2 + c_1 x}$$

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

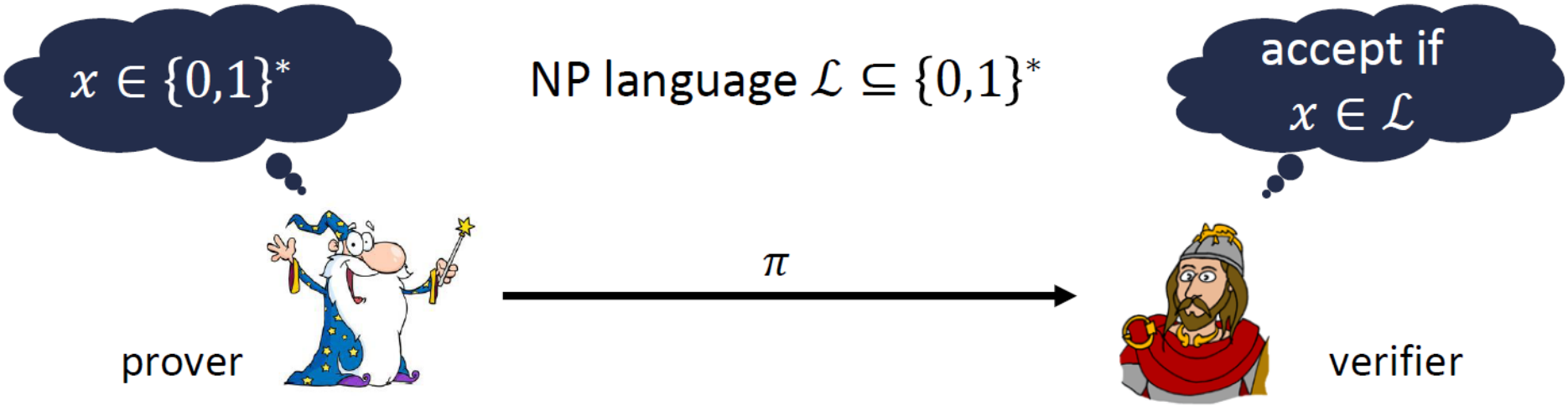
Thus, extractor succeeds with **overwhelming probability**.

Σ-protocols

Problem: Prover and verifier must be online at the same time. Because the protocol requires interaction!

$\Rightarrow$ Solution: Make the protocols **non-interactive**.

Non-Interactive Zero-Knowledge



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

"Honest prover convinces honest verifier of true statements"

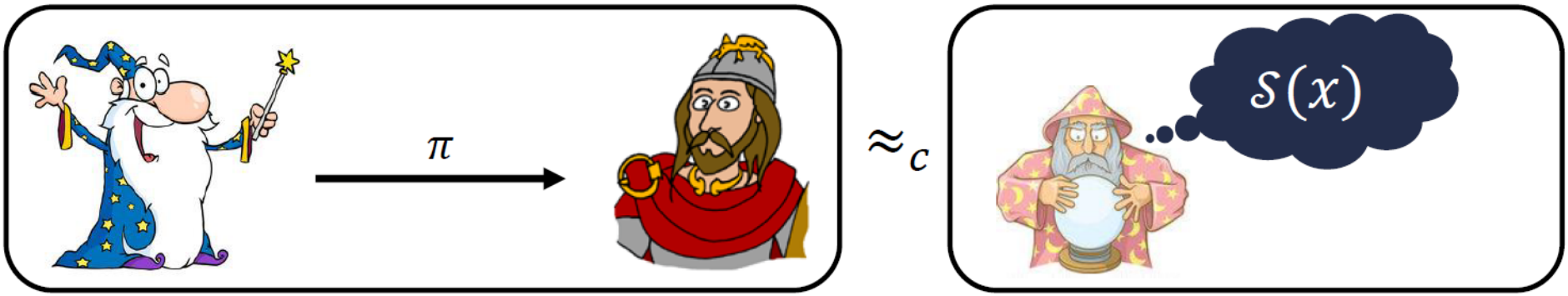
Soundness: $\forall x \notin \mathcal{L},\ \forall P^* : \Pr[\langle P^*, V \rangle(x) = \text{accept}] \leq \varepsilon$

"No prover can convince honest verifier of false statement"

can consider both computational and statistical variants

Non-Interactive Zero-Knowledge

NP language $\mathcal{L}$



real distribution ideal distribution

Zero-Knowledge: for all efficient verifiers V^*, there exists an efficient simulator $\mathcal{S}$ where

$$\forall x \in \mathcal{L} : \langle P, V^* \rangle(x) \approx \mathcal{S}(x)$$

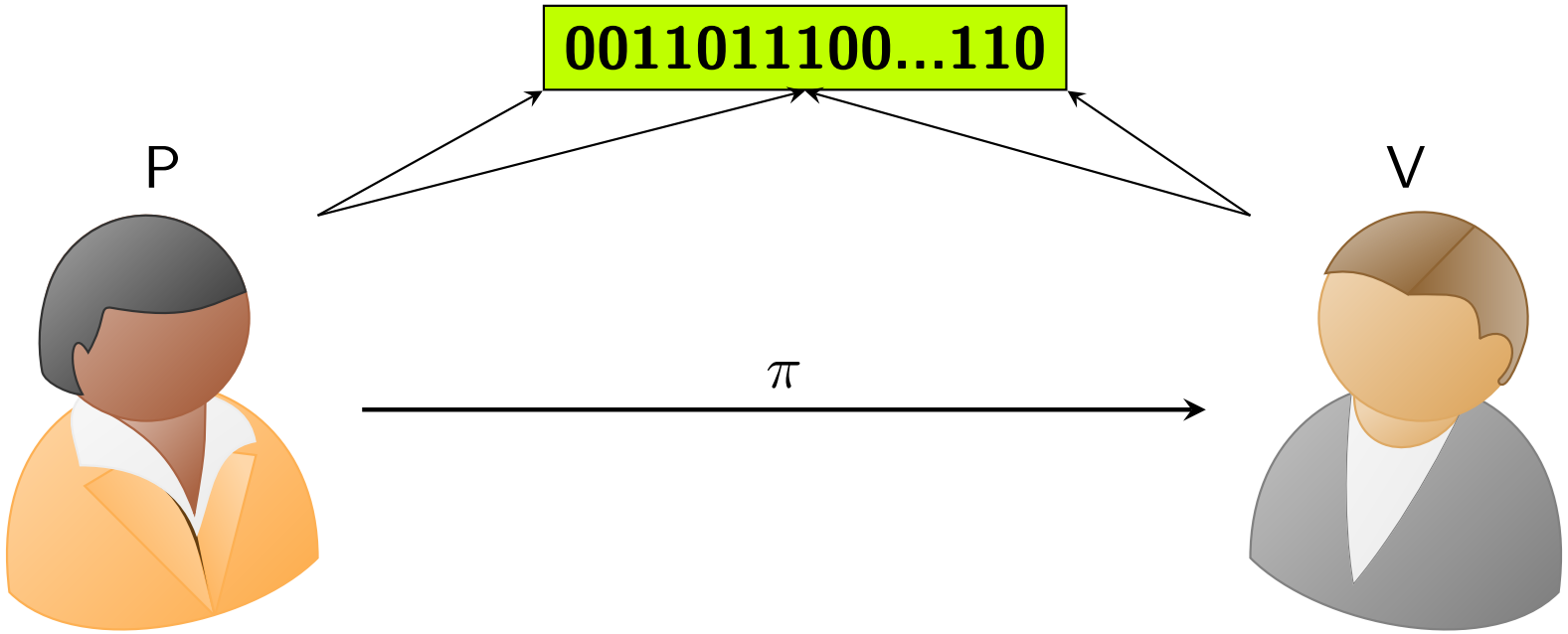
can consider both computational and statistical variants

Non-Interactive Zero-Knowledge

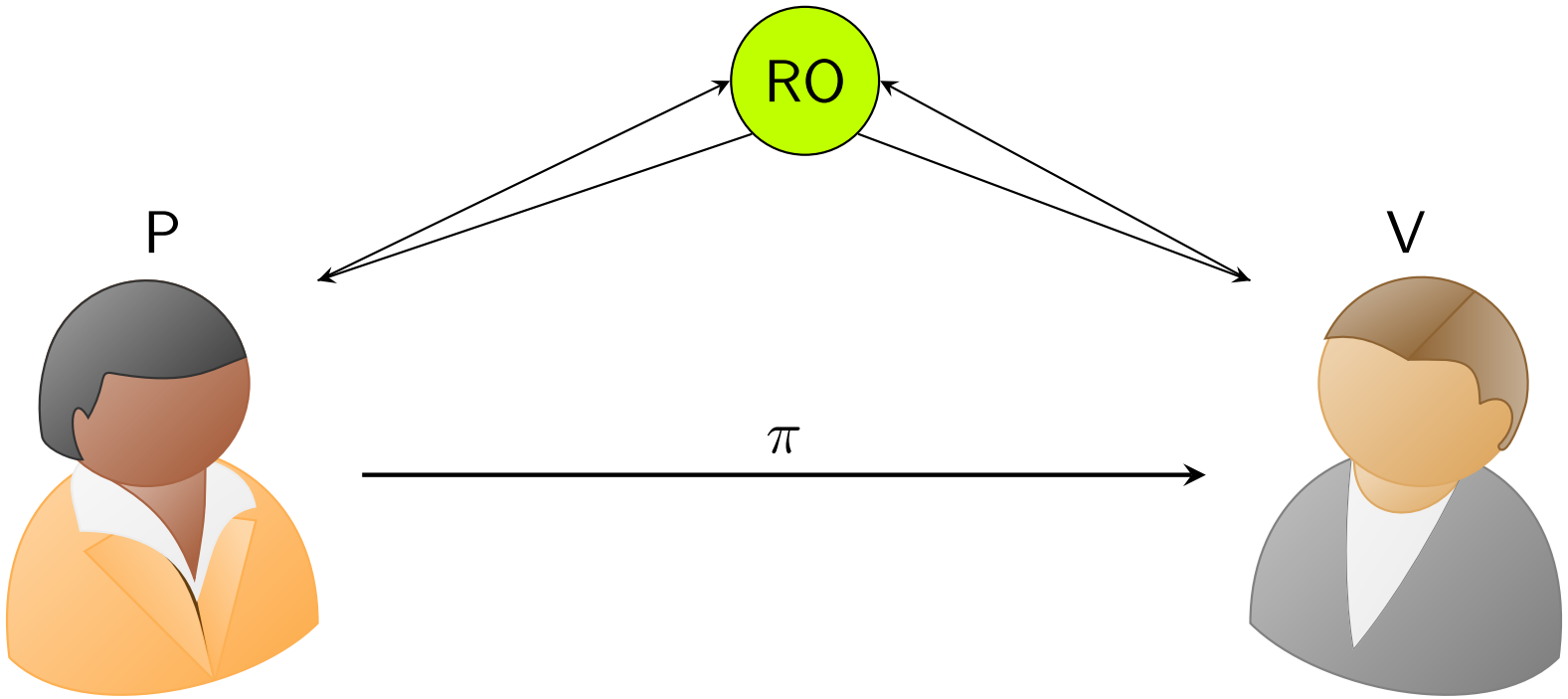
A zero-knowledge proof system where the proof is a single message from the prover to the verifier.
It is important because interaction is **expensive** in practice.

- Unfortunately, NIZK proofs are only possible for sufficiently easy languages. ($\mathcal{BPP}$ languages)
  -> The reason is that we can use the simulator to decide the language:
  1. if $x \in L$, $\pi = S(x)$ should be accepted by the verifier. (by ZK property)
  2. if $x \notin L$, $\pi = S(x)$ should not be accepted by verifier. (by soundness)

- **Common Reference/Random String Model**:

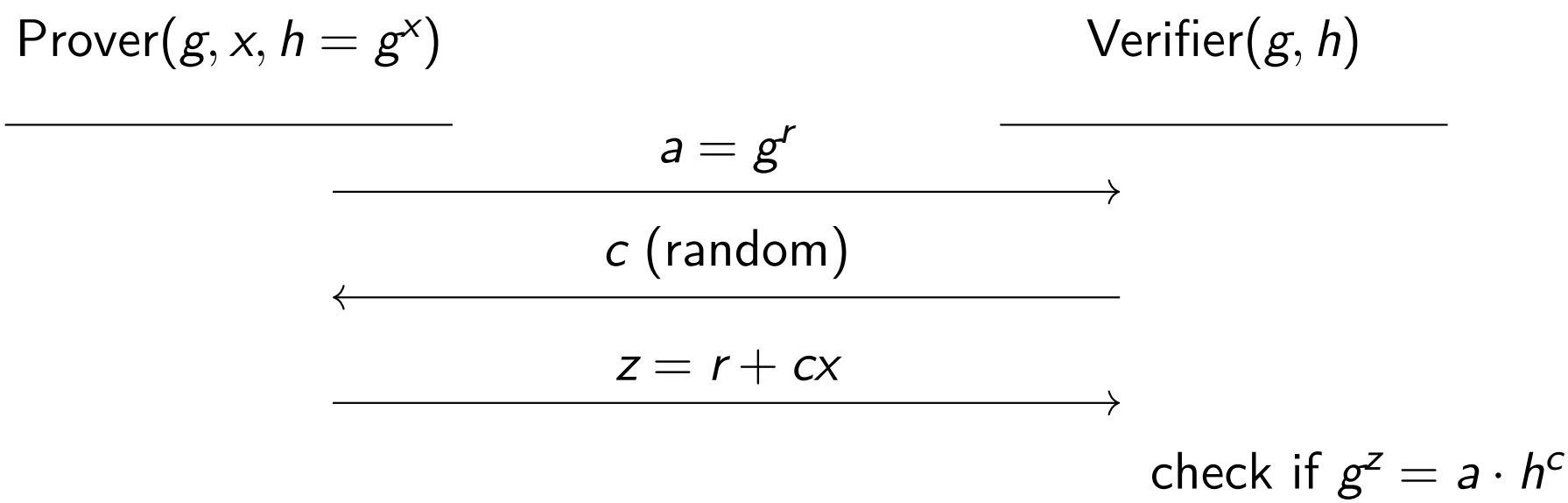


- **Random Oracle Model**:



Fiat-Shamir Transform

- **Fiat-Shamir transform**: From Σ-protocols to NIZK in RO Model
  Recall Schnorr's protocol (from Σ-protocols):



-> The idea is using a hash function (random oracle) H instead of verifier's challenge. So prover can computes it **on its own**.

$$\Rightarrow c = H(g, h, a)$$

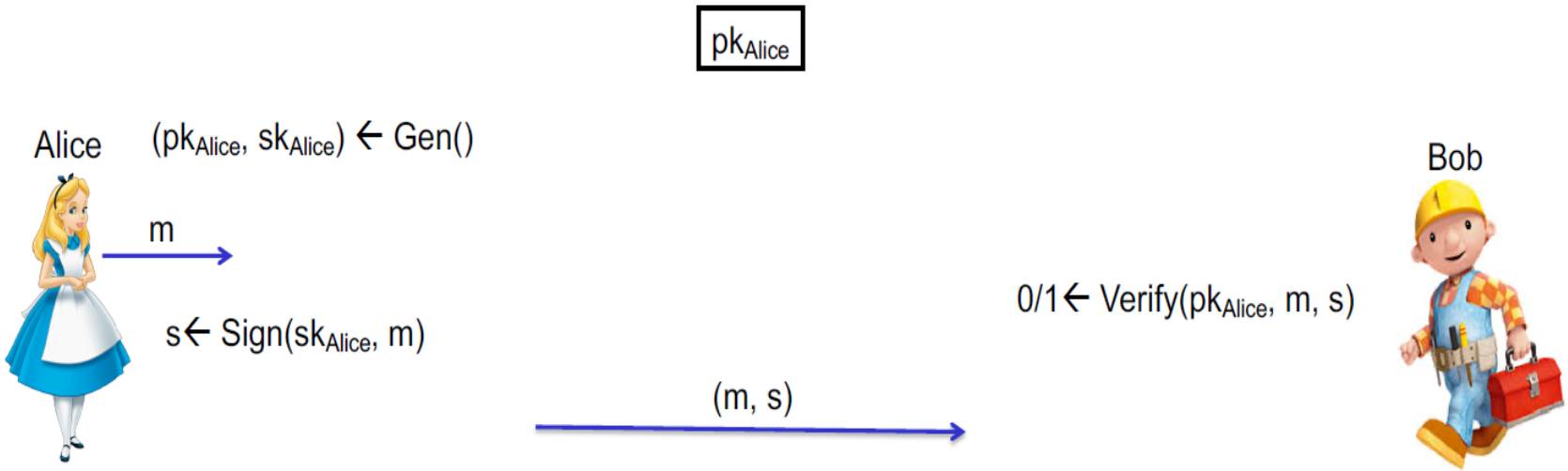
Fiat-Shamir Transform

- Fiat-Shamir transform on Schnorr's protocol (Schnorr's signature) from discrete log problem in RO model:
  - **Setup:** Sample $x \xleftarrow{R} \mathbb{Z}_p$, and set $vk = (g, h = g^x)$ and $sk = x$.
  - **Sign**(sk, m): Sample $r \xleftarrow{R} \mathbb{Z}_p$ and set $a \leftarrow g^r$. Compute $c \leftarrow H(g, h, a, m)$ and set $z \leftarrow r + cx$. Output $\sigma = (u, z)$.
  - **Verify**(vk, m, σ): Parse $\sigma = (u, z)$ and $vk = h$, compute $c \leftarrow H(g, h, a, m)$ and accept if $g^z = a \cdot h^c$.

-> Signature is a NIZK proof of knowledge of discrete log of h. And security follows from the security of Schnorr's protocol (together with Fiat-Shamir).

Fiat-Shamir Transform

- More generally, any Σ-protocol can be used to build a signature scheme using the Fiat-Shamir heuristic (by using the message to derive the challenge via RO).
- To recall digital signatures:



Thank you!