The **Brave New World** of Global **Generic Groups** and **UC**-Secure Zero-Overhead **SNARKs**

Jan Bobolz — The University of Edinburgh, ZK Lab

Pooya Farshim — Input | Output, Durham University

Markulf Kohlweiss — The University of Edinburgh / Input | Output, ZK Lab

Akira Takahashi — J.P.Morgan AlgoCRYPT CoE, AI Research

Slides in courtesy of Jan Bobolz TCC 2024 ia.cr/2024/818

Prove$(x, w) \to \pi$

zkSNARKs

Verify$(x, \pi) \to b$

Zero-Knowledge

π does not reveal *any information* about w.

Efficient **Simulator**

$\mathcal{S}(x) \to \pi$

Malicious prover can run $\mathcal{S}$ to compute π *without knowing w*

Proof of knowledge

In order to compute valid π, prover must *know* w.

Efficient **Extractor**

$\mathcal{E}(x, \pi) \to w$

Malicious verifier can run $\mathcal{E}$ to learn *full information on w*

Simulator and extractor need a

superpower

that malicious provers/verifiers don’t have.

Prove$(x, w) \rightarrow \pi$

zkSNARKs

Verify$(x, \pi) \rightarrow b$

Efficient **Simulator**
$\mathcal{S}(x) \rightarrow \pi$

$\mathcal{A}^{\mathcal{S}}$

x', π'

Efficient **Extractor**
$\mathcal{E}(x', \pi') \rightarrow w$

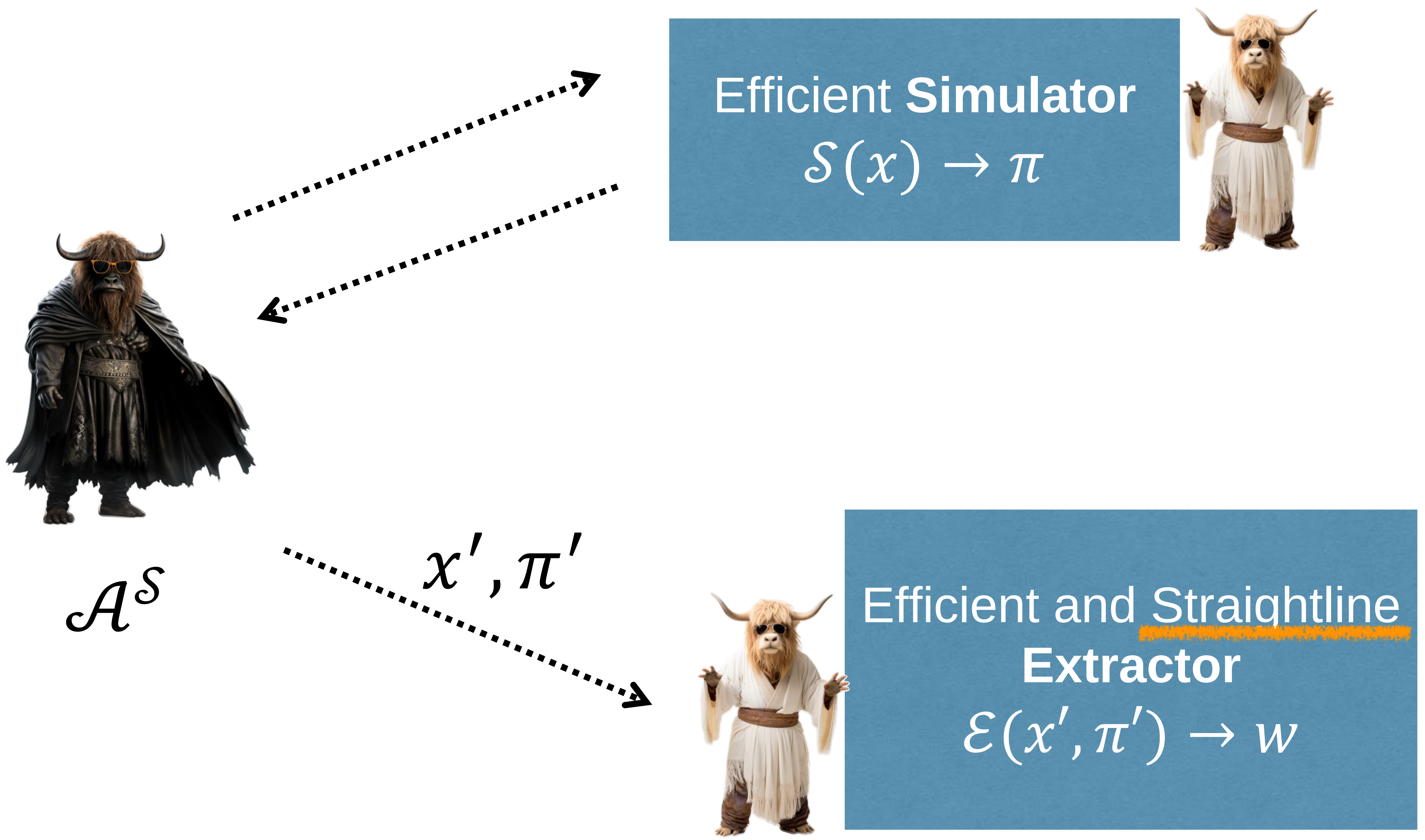
Simulation Extractability

In order to compute valid π, prover must *know* w, even after observing simulated proofs

Sim-Ext is often a precondition of

UC-secure NIZK

UC-Secure zkSNARKs

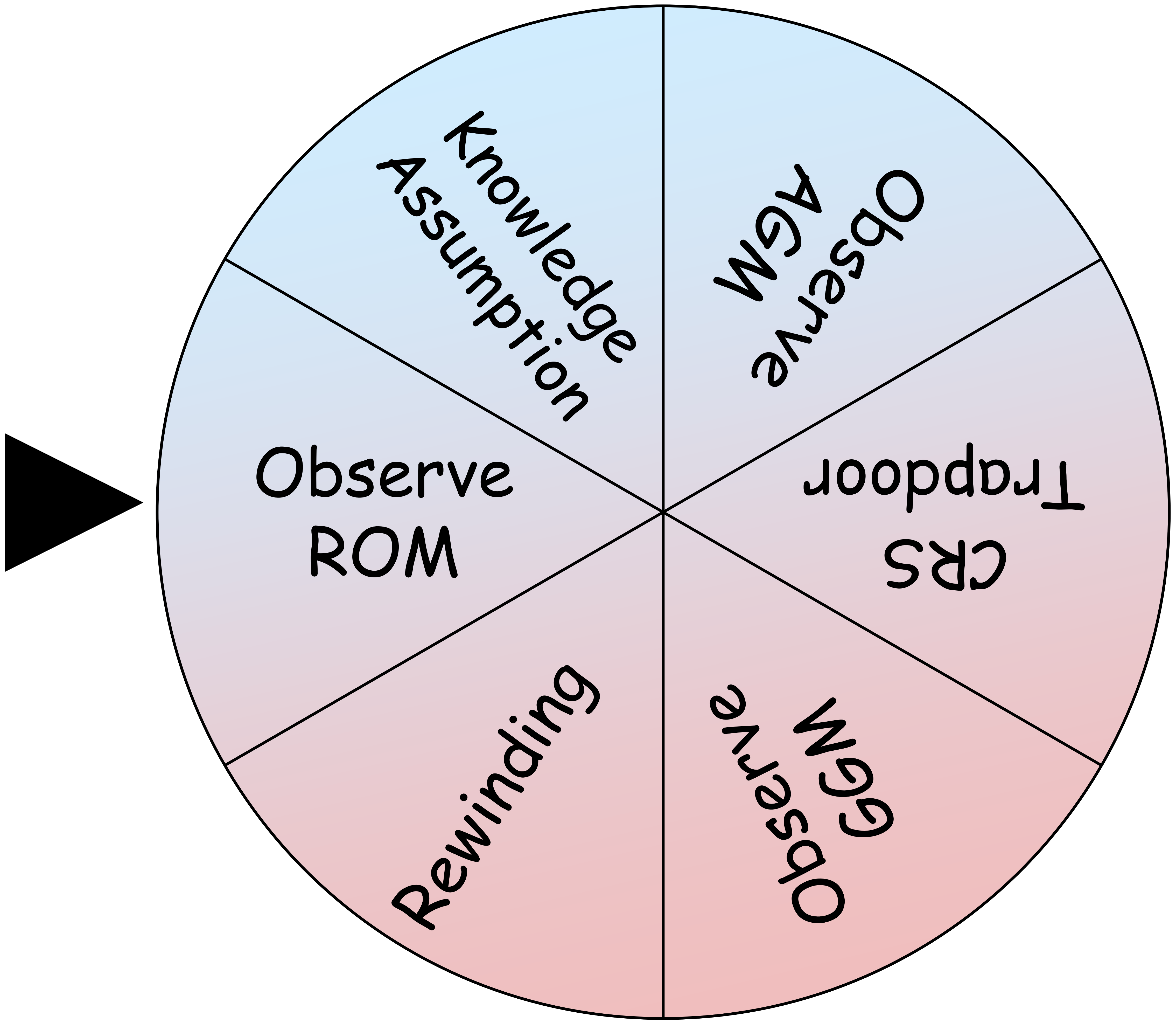


- Generic compilers turning NIZK with standalone proofs of security into UC-secure ones: [KZM+15] [ARS20] [BS21] [LR22] [CSW22] [AGRS23] [GKO+23]
- Incur overhead in proof sizes and/or prover time!
- Exception: [CF24] for hash-based, already-straightline-extractable SNARKs (previous talk)

Can we design a group-based idealized model allowing for

UC-secure SNARKs without overhead?

Let's spin the PoK wheel 🎰

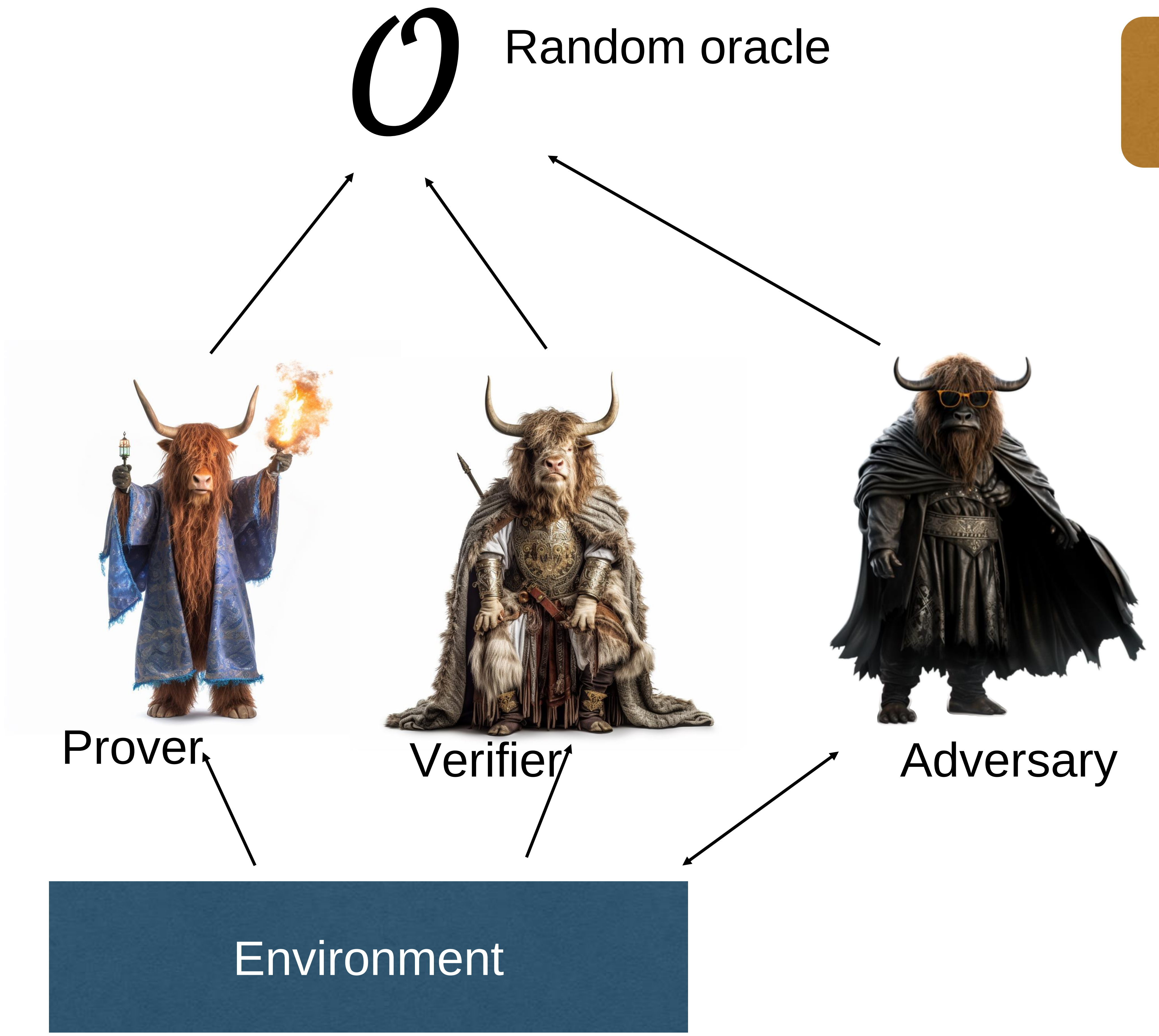


The UC RO hybrid model

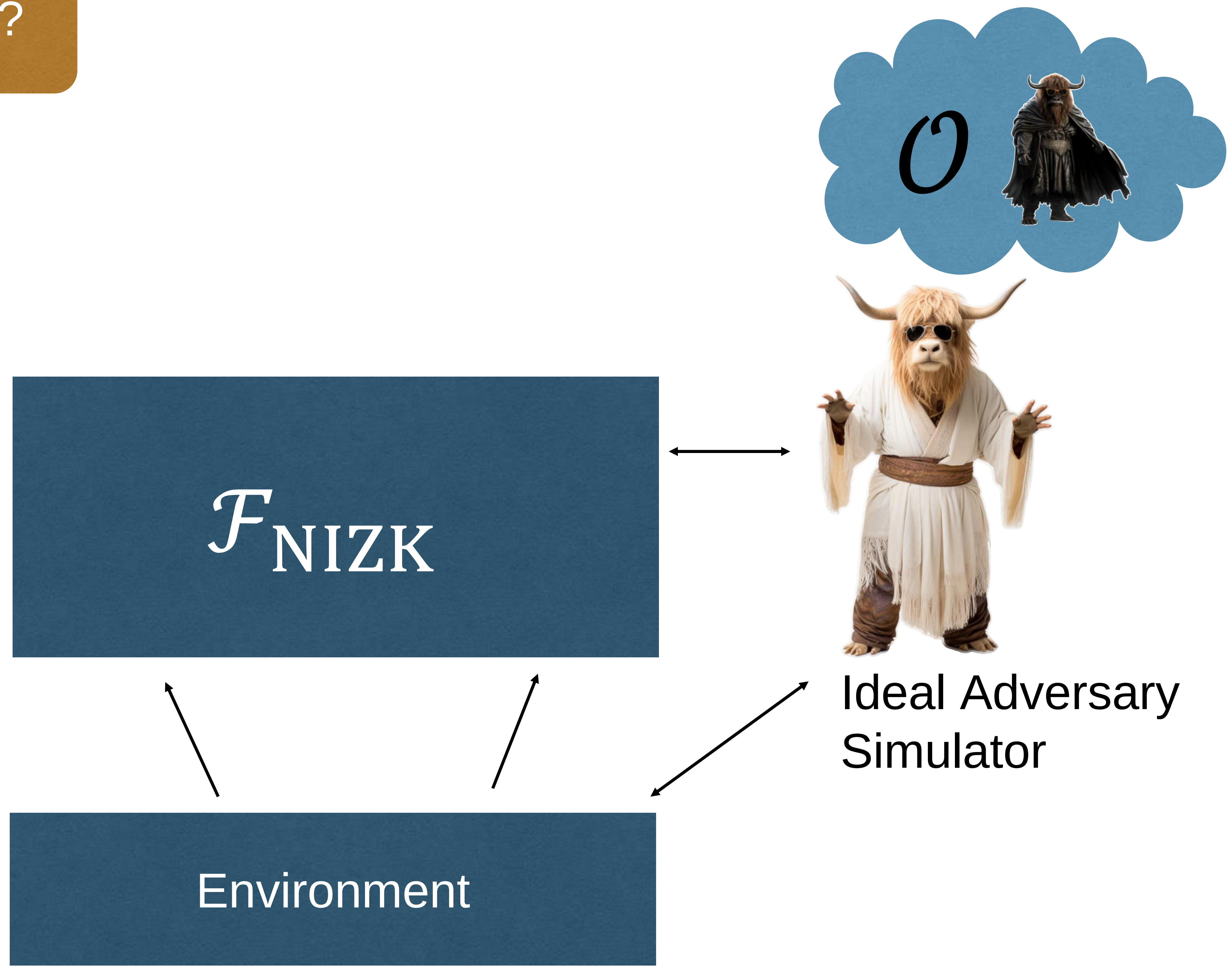
eval(x):
If $Q(x) = \perp$:
  $Q(x) \leftarrow \{0,1\}^n$
return $Q(x)$

Real world

Ideal world

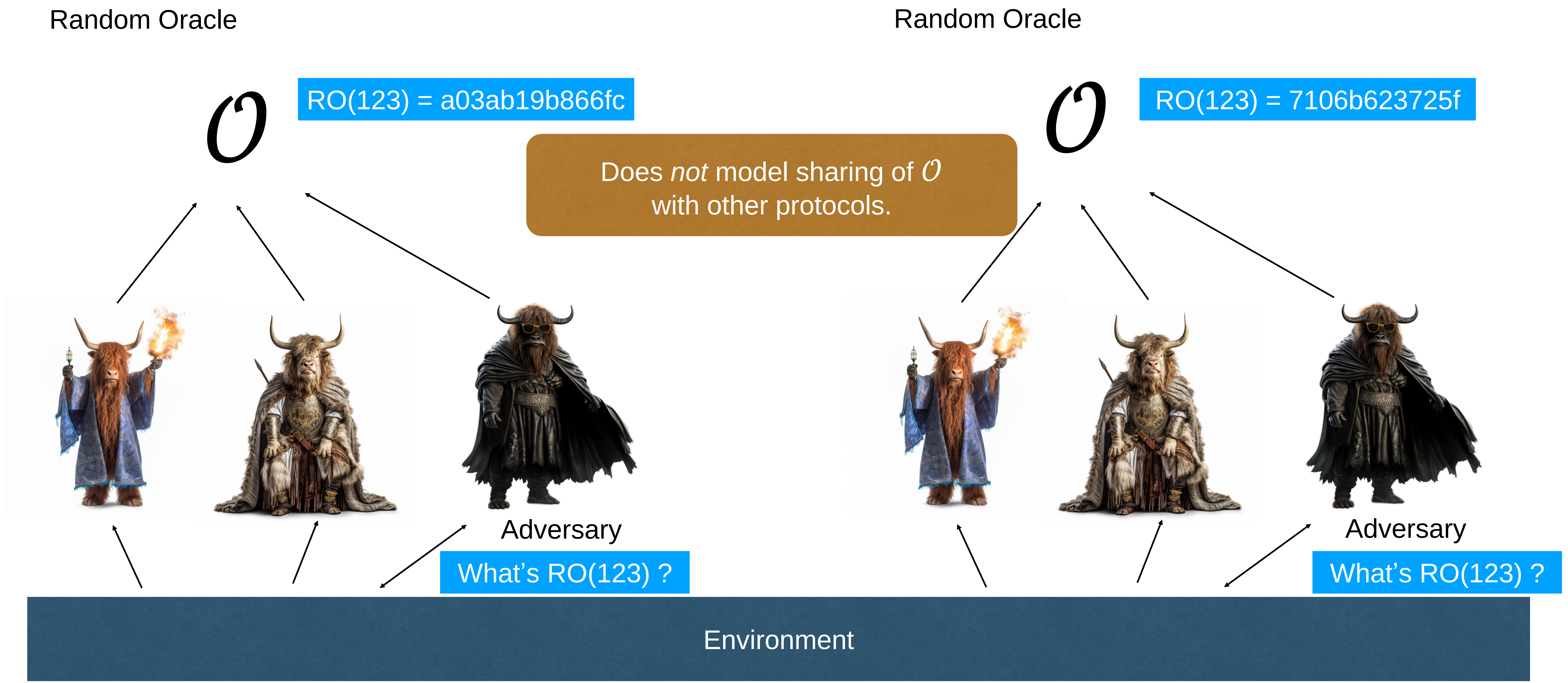


Is this a good model?



Is this a good model?

Let’s have two sessions of the protocol



Superpower 1:
Global Random Oracles

From

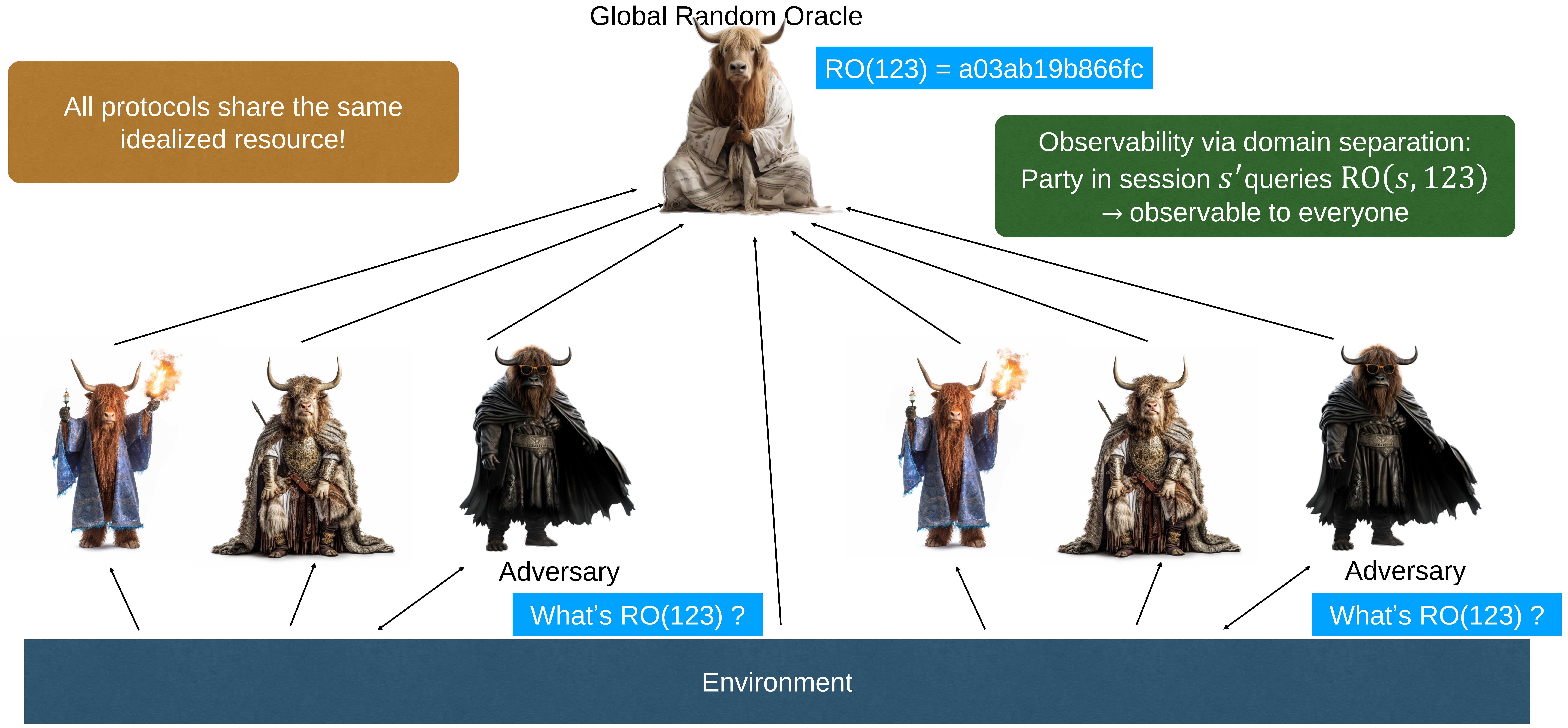
Practical UC security
with a global random oracle

Ran Canetti, Abhishek Jain, Alessandra Scafuro
CCS 2014

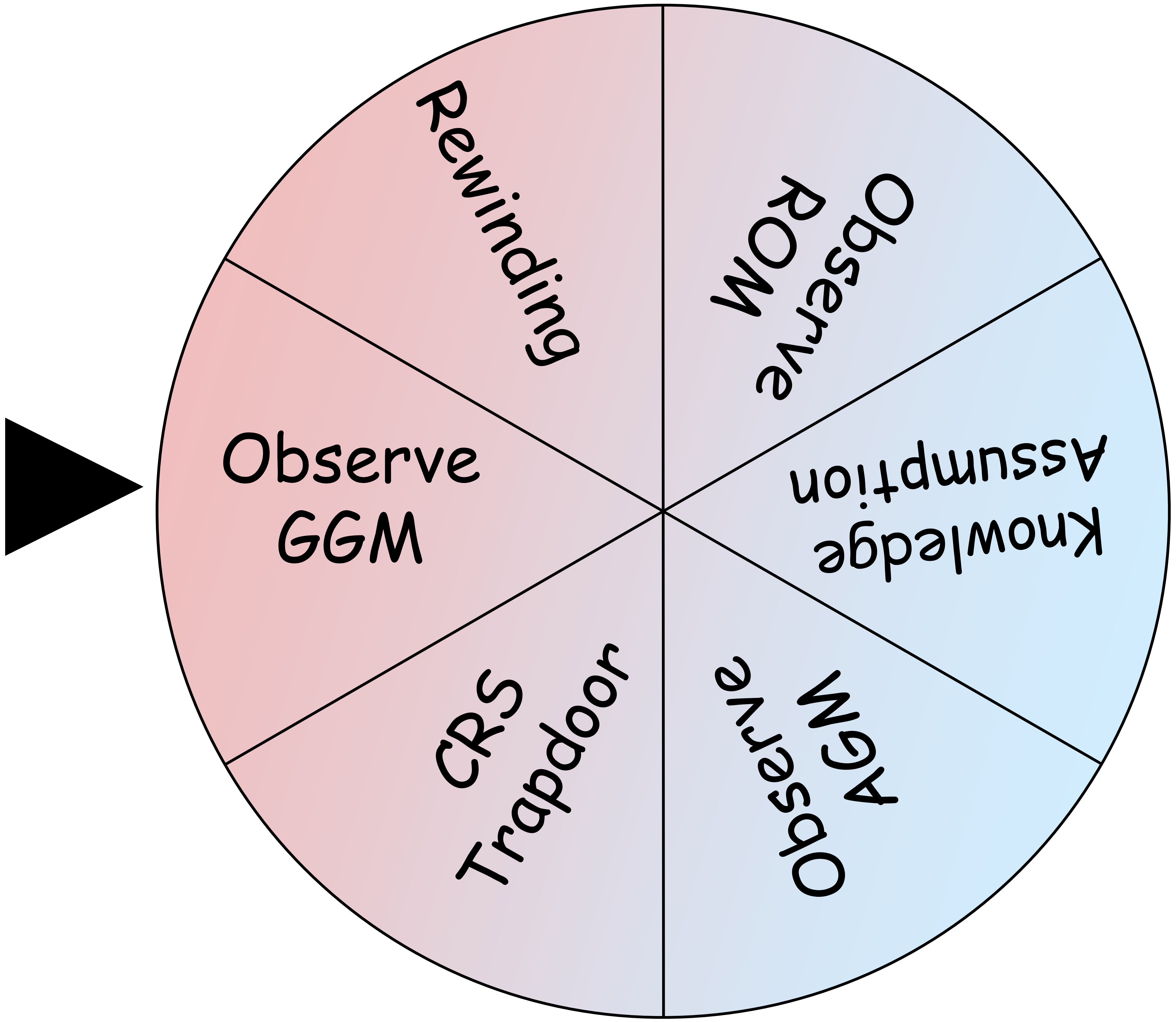
The Wonderful World
of Global Random Oracles

Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven
Eurocrypt 2018

The better model: Global ROM



Let's spin the PoK wheel 🎰



The generic group model

- 🎯 Goal: model “**idealized**” group with no “extra” structure (just group operations[, pairing]).
  - Similar to random oracles, which model “idealized” hash function with no structure.
- 💡 Idea: group elements get ***random* encoding** (= no structure), but **oracle enables group ops**.
- 👀 Corollary: oracle *sees* all group ops.
-  PoK Extractor recovers witness from observed group ops

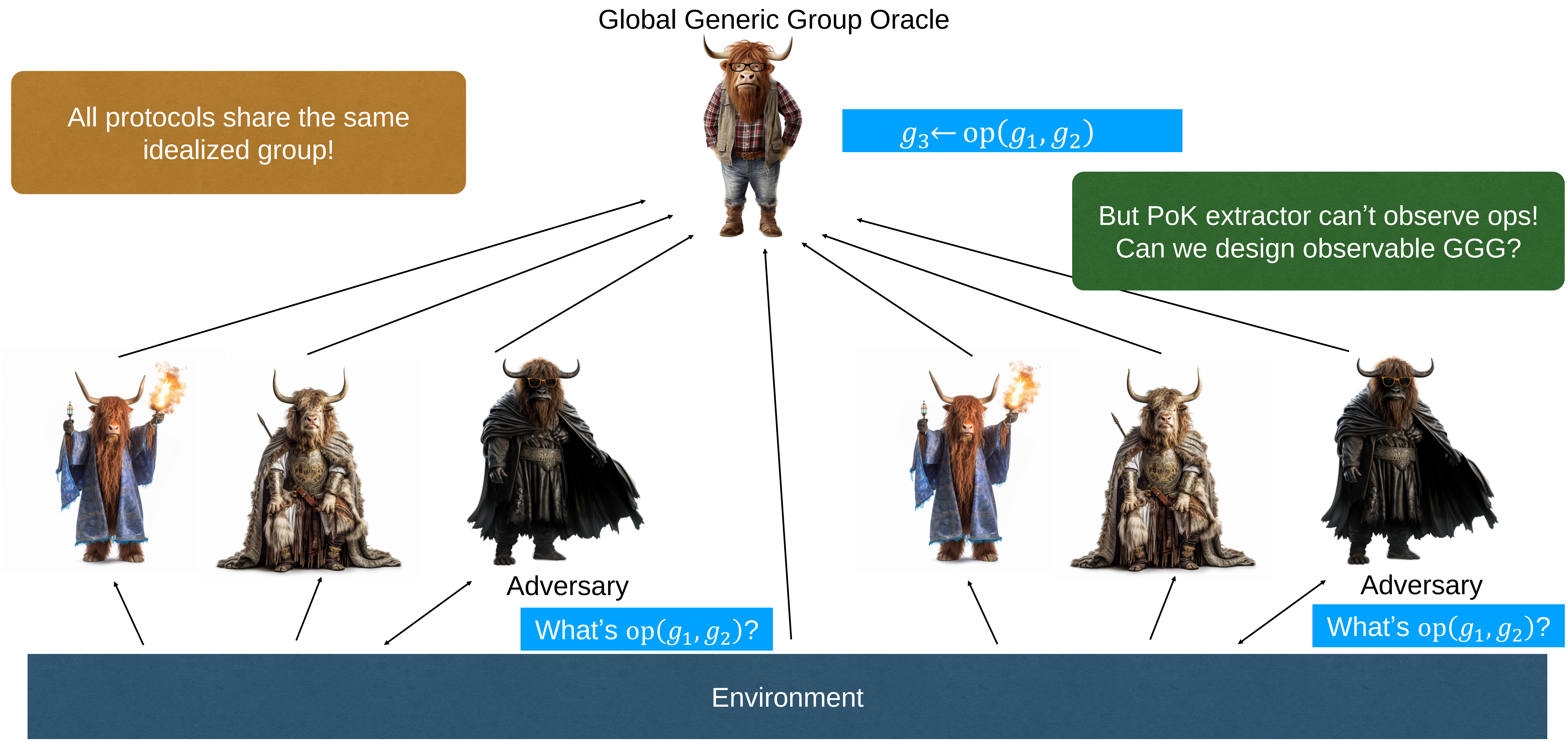
The generic group model

$\mathcal{O}$ =

`private` random injective $\tau: \mathbb{G} \rightarrow S$
`public` generator g

op(g_1, g_2):
return $\tau(\tau^{-1}(g_1) + \tau^{-1}(g_2))$

First Step: “Strict” Global GGM in UC



Example: Groth16 PoK in GGM

Extractor

observe()

$$A = \sum_{i=0}^{m} a_i[u_i] + [\alpha] + r[\delta]$$

$$B = \sum_{i=0}^{m} a_i[v_i] + [\beta] + r'[\delta]$$

$$C = \cdots$$

Prover

Verifier

[Check some pairing equations on A,B,C]

CRS: group elements $[u_i], [\alpha], [\beta], [\delta]$

Witness: wire values $a_i \in \mathbb{Z}_p$

Design challenges

- 👷 Requirements:
  - 👁️ Simulator/Extractor **must see** group operations made by environment
    - Required to extract
  - Environment **must not see** what group operations simulator makes
    - Would immediately reveal that we simulate
- ⚠️ First glance: **Impossible**
- 💡 Do **partial observability** via **domain separation**

Design challenges

- 👷 Requirements:
  - 👁️ Simulator/Extractor **must see** (relevant) group operations made by environment
    - Required to extract
  - Environment **must not see** what group operations simulator makes
    - Would immediately reveal that we simulate
- ⚠️ First glance: **Impossible**
- 💡 Do **partial observability** via **domain separation**

Observation rules (intuition)

- 📜 Every session s gets its own group generator h_s
- 👍 Legal/unobservable: Session s operates on h_s
- 👎 Illegal/observable: Session s' operates on h_s

G-oGG: Observable Global Generic Group (Simplified)

`private` random inj. $\tau: \mathbb{G} \to S$
`public` rnd generator h_s for each session s
`public` poly variable X_s for gen of each session s
`private` representation $R[e]$ for each $e \in S$, initially $R[h_s] = X_s$

<u>op(g_1, g_2):</u>
$s = caller\ session$
$result = \tau(\tau^{-1}(g_1) + \tau^{-1}(g_2))$
$R[result] = R[g_1] + R[g_2]$ //bookkeep sum of polynomials
if $R[result] \notin \mathbb{Z}_p[X_s]$: //invalid in caller session
  Add $(g_1, g_2, result)$ to public observation list
return $result$

<u>Intuition</u>
Cross-session operations are observable

<u>Example ops with caller session s</u>

- $17X_s \mathrm{op} X_{s'}$ observable
- $(17X_s + 3X_{s'}) \mathrm{op} X_s$ observable
- $17X_s \mathrm{op} 4X_s$ unobservable
- $(17X_s + 0X_{s'}) \mathrm{op} X_s$ unobservable

Actual G-oGG

- Multiple generators per session
- Oblivious Sampling
- 💫 Pairing operations

Summary: ROM vs GGM in UC

Local ROM: bad model 🙁 Both sessions use SHA-3, why am I getting different hashes?	**Local GGM**: bad model 🙁 Both sessions use BLS12-381, why are elements incompatible?
Global ROM: **lose observability**. Remodel. Environment/other protocols can access global ROM without going through the simulator.	*Global* GGM: **lose observability**. Remodel. Environment/other protocols can access global GGM without going through the simulator.
Domain separation: $\mathrm{RO}(s, x)$ is "valid/in-session" iff caller is in session s. Invalid queries are observable.	**Domain separation**: $\mathrm{op}(g_1, g_2)$ is "valid/in-session" iff g_1, g_2 are based on caller session's generator h_s

ZK: honest parties only make "valid" unobservable queries within their domain.
PoK: when environment / protocol in session $s' \neq s$ queries related to domain s, it's observable.

Groth16 proof challenges

Extraction

Idea
Extract dlog representation of proof elements

Challenge
Cannot observe *everything* (only *my* session's generator(s))

💡 Solution
Argue that valid proofs cannot contain foreign generators

Simulation

Idea
Use CRS trapdoor to generate proofs without witness

Challenge
Prover/Simulator GGM ops must not be observable

💡 Solution
Prover/simulator only operates on CRS elements

Takeaways

- New design of global generic groups in UC
- 👀 To prove SNARKs UC-secure in GGGM, we need to explicitly **model observability**
  - **Not trivial!**
- Unlike UC-AGM [ABK+21], we introduce a **global** GG functionality while the original UC(GS) framework remains unchanged
- Case study: Get Groth16 SNARK in UC (against static corruptions) without modifications

Algebraic Adversaries in the Universal Composability Framework. Michel Abdalla, Manuel Barbosa, Jonathan Katz, Julian Loss, and Jiayu Xu. Asiacrypt'21