Groth16 is UC-secure

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

ia.cr/2024/818

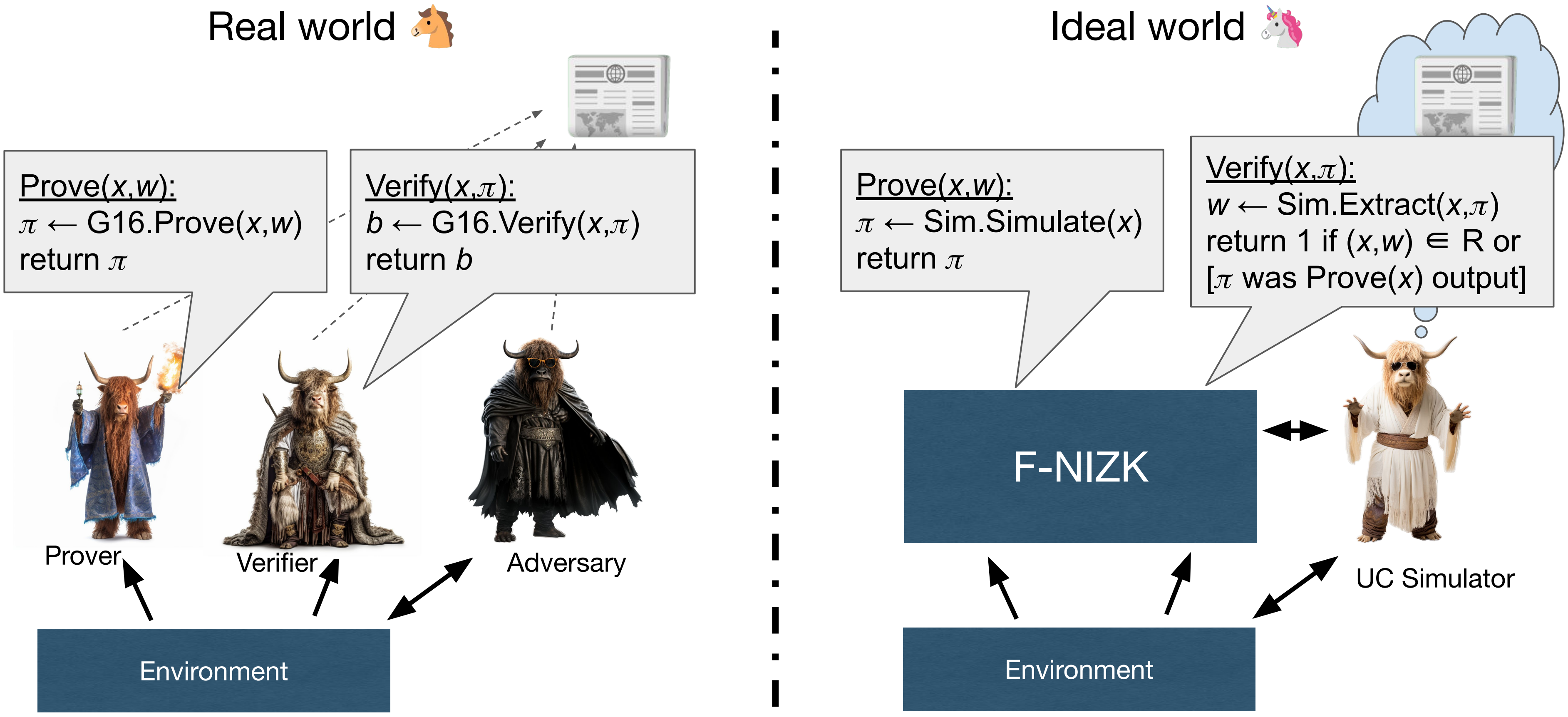
Jan Bobolz — The University of Edinburgh, ZK Lab

Pooya Farshim — Input | Output, Durham University

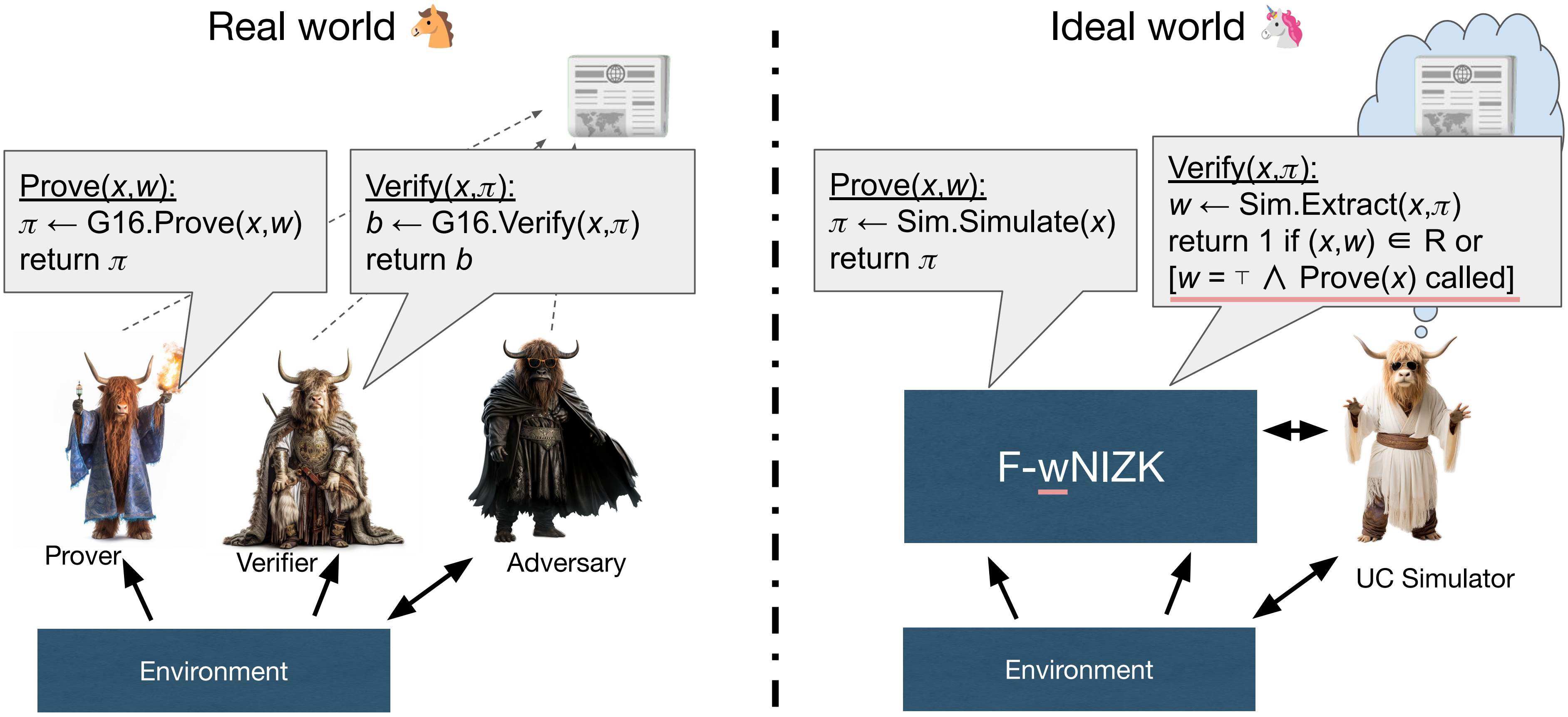
Markulf Kohlweiss — Input | Output, The University of Edinburgh, ZK Lab

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

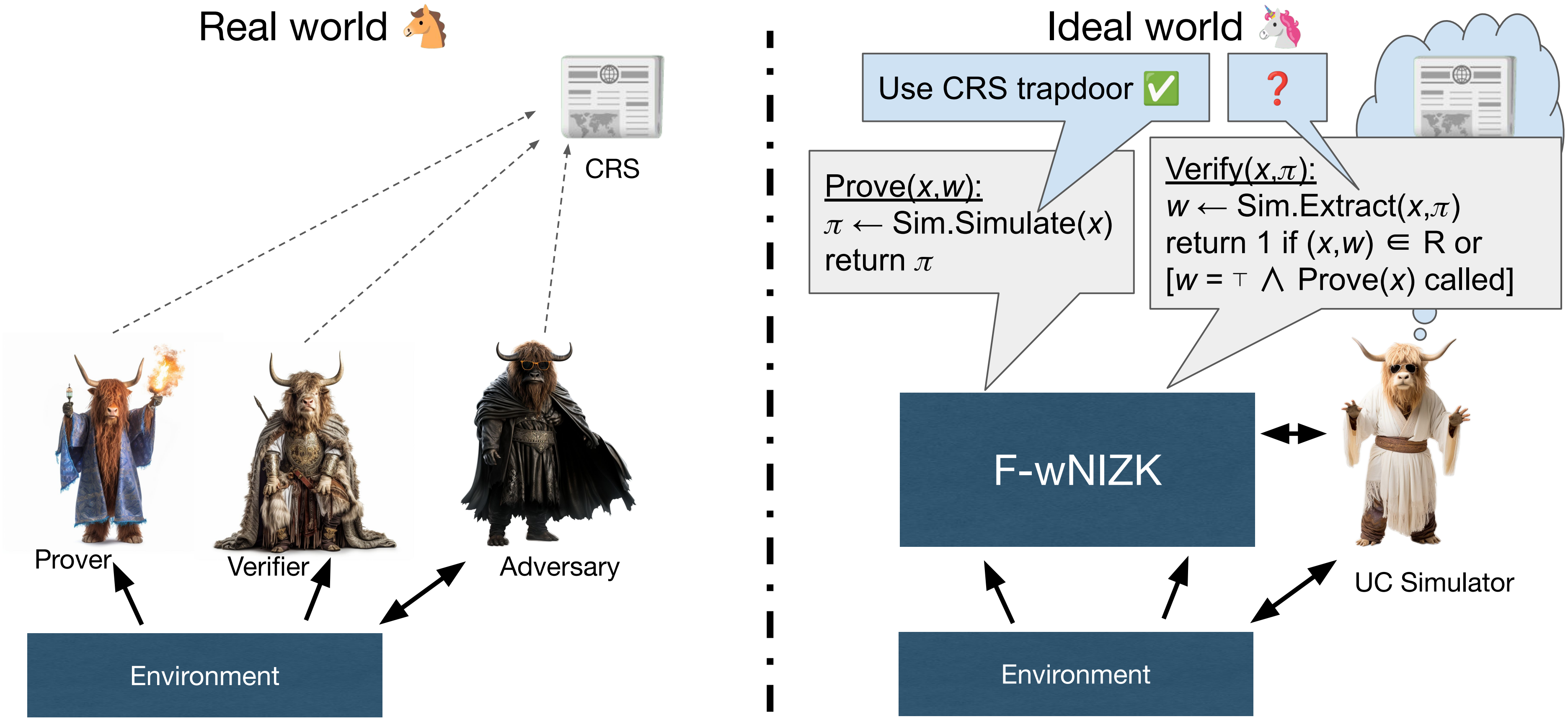
UC modeling of NIZKs



UC modeling of NIZKs



UC modeling of NIZKs



Extraction

Extractor advantage	Groth16 extraction	UC modeling
🎲 **GGM** Generic group model	✅ [Gro16]	❌ (only folklore *local* F-GGM)
🧮 **AGM** Algebraic group model	✅ ~[FLK18, BKSV21]	❌ UC-AGM [ABKLX21] Unsuitable for NIZKs
🧠 **KA** Knowledge assumptions	✅ ~[GM17, BFHK23]	❌ In CC framework [KKK21] Unsuitable, problems like AGM
📰 **CRS** Common reference string	📦 via compiler (like C∅C∅) [KZMQCPRsS15], […]	✅ F-CRS hybrid model
🔮 **ROM** Random oracle model	📦 via compiler [GKOPTT23]	✅ Global observable RO [CDGLN18] (e.g., [CF24])
⏪ **Rewinding**	❌ (❓)	❌ (does not compose)

The generic group model

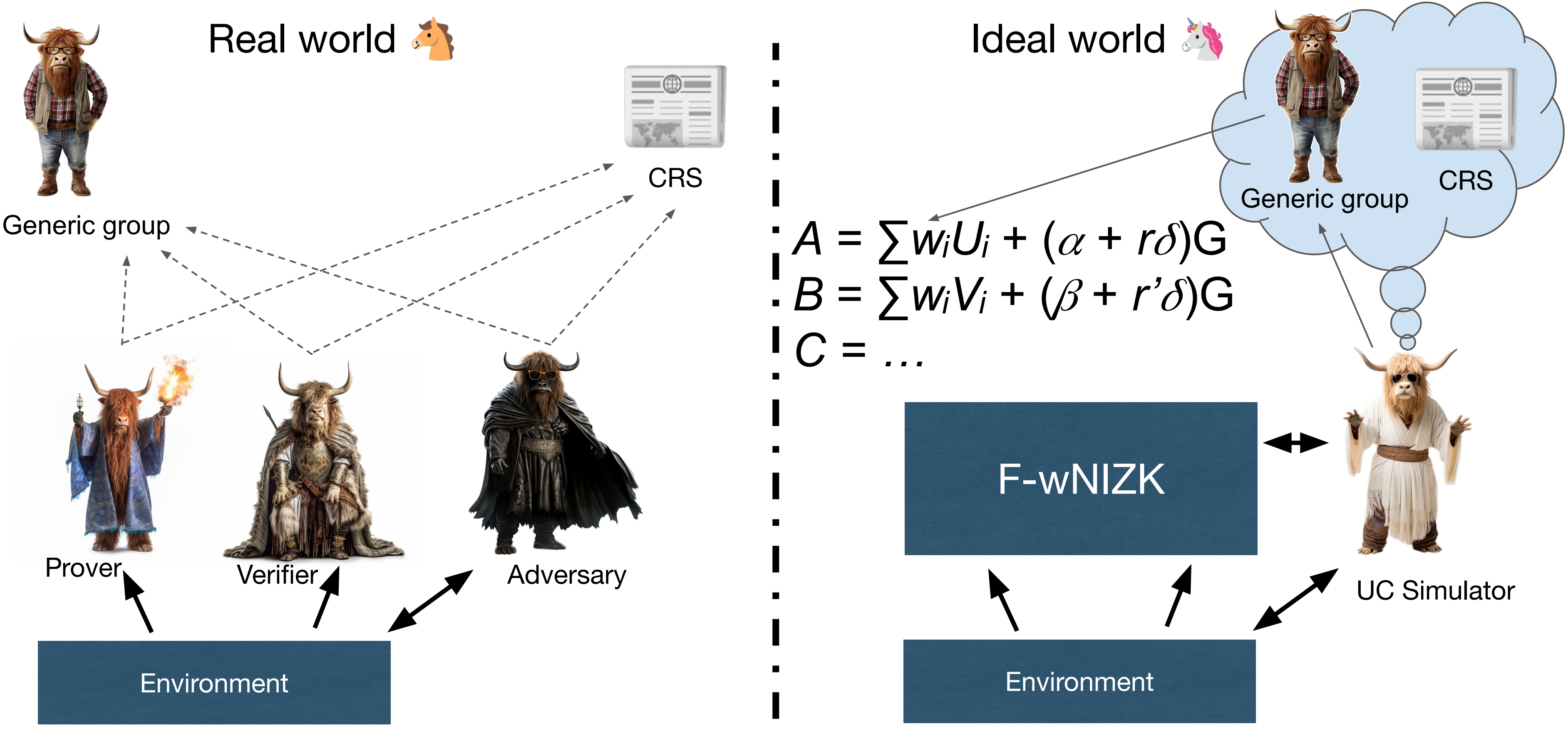
`private` random injective τ: $\mathbb{G} \rightarrow$ S

`public` generator $g \in$ S

op(g_1, g_2):

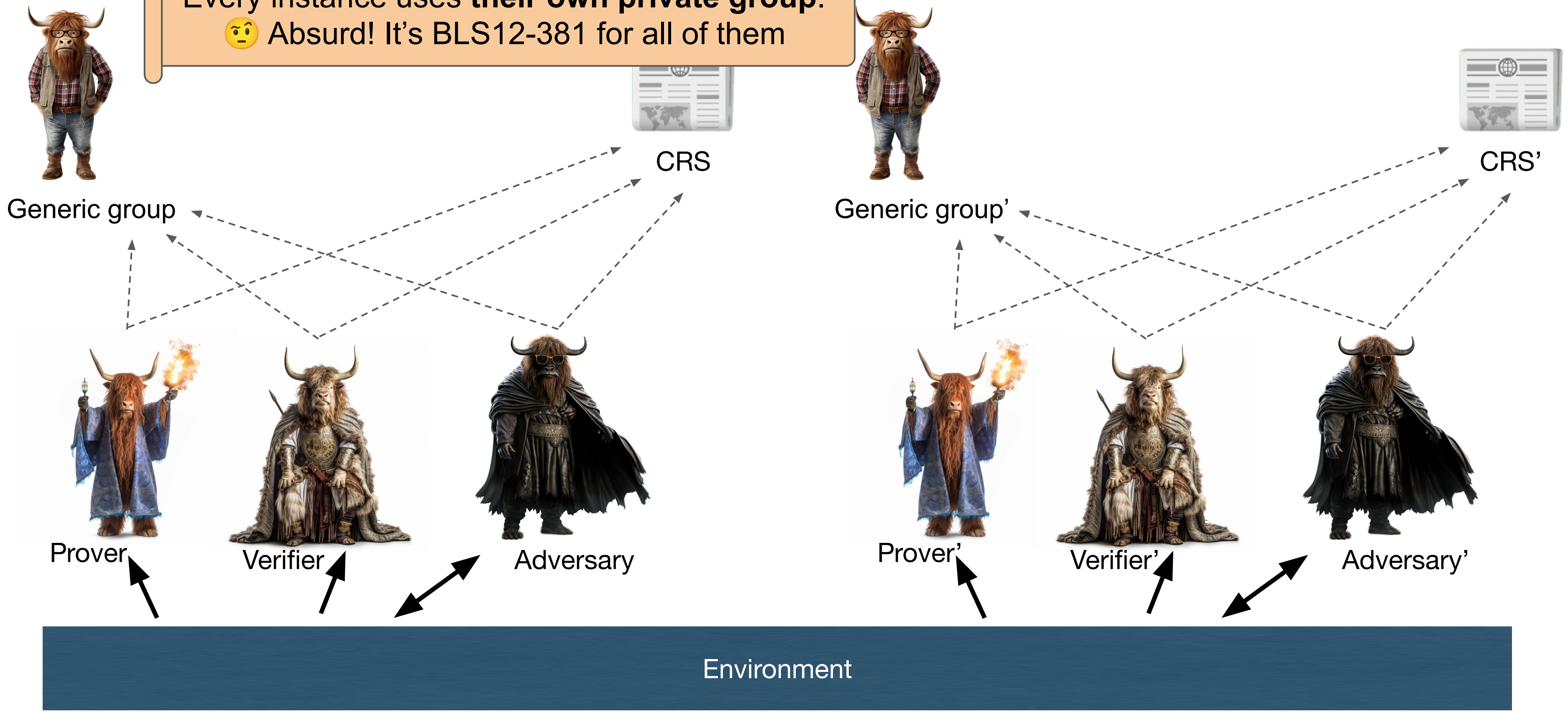
    return $\tau(\tau^{-1}(g_1) + \tau^{-1}(g_2))$

(Local) generic groups in UC



(Local) generic groups in UC

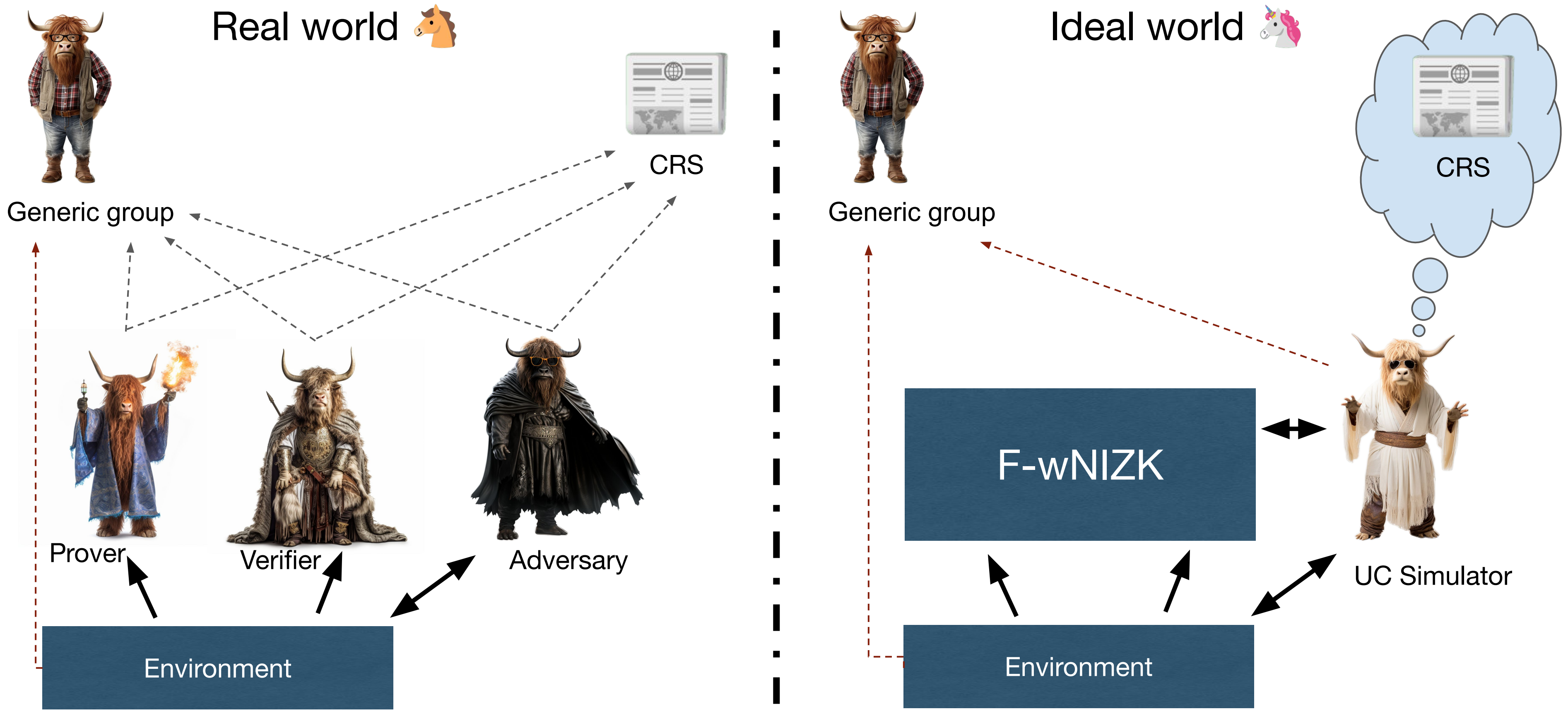
Every instance uses **their own private group**.
🤨 Absurd! It's BLS12-381 for all of them



Extraction

Extractor advantage	Groth16 extraction	UC modeling
🎲 **GGM** Generic group model	✅ [Gro16]	✅ Global observable generic group functionality (G-oGG)
🧮 **AGM** Algebraic group model	✅ ~[FLK18, BKSV21]	❌ UC-AGM [ABKLX21] Unsuitable for NIZKs
🧠 **KA** Knowledge assumptions	✅ ~[GM17, BFHK23]	❌ In CC framework [KKK21] Unsuitable, problems like AGM
📰 **CRS** Common reference string	📦 via compiler (like C∅C∅) [KZMQCPRsS15], […]	✅ F-CRS hybrid model
🔮 **ROM** Random oracle model	📦 via compiler [GKOPTT23]	✅ Global observable RO [CDGLN18] (e.g., [CF24])
⏪ **Rewinding**	❌ (❓)	❌ (does not compose)

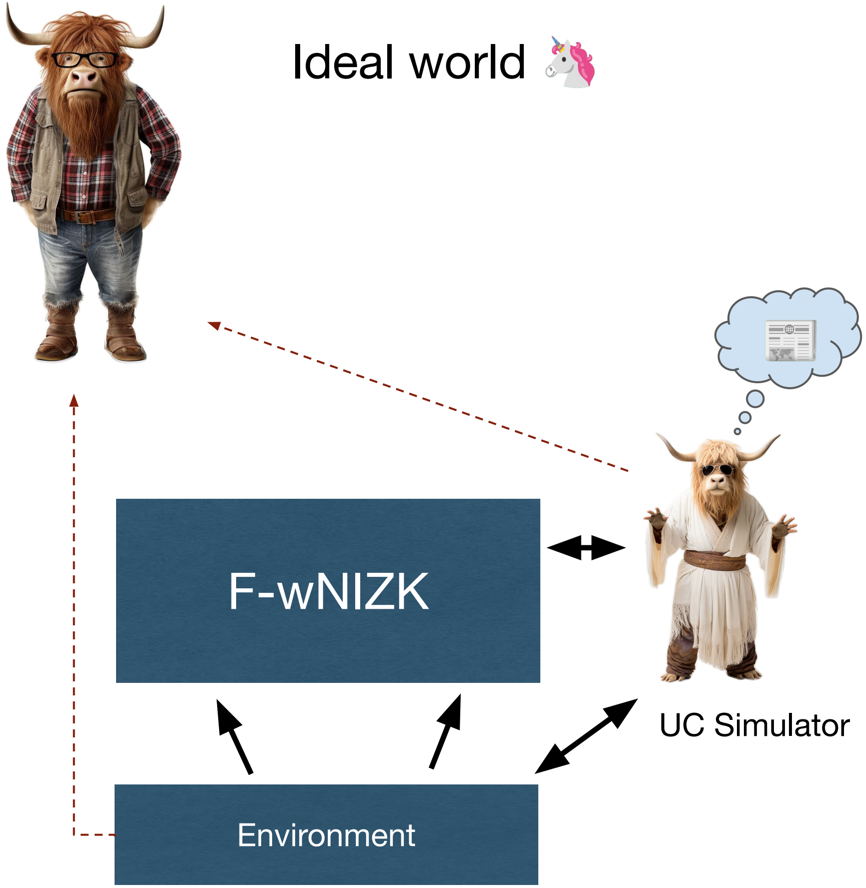
Global generic groups in UC



Global generic groups in UC

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

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



🚨 **Issue**: UC-simulator does not see the environment's group operations.
⚔️ **Attack**: Env honestly computes proof π, then calls Verify. Extraction fails.

Fully observable global generic groups in UC

`private` random injective $\tau: \mathbb{G} \to \mathsf{S}$

`public` generator $g \in \mathsf{S}$

`public` observation list *Obs*

$\mathrm{op}(g_1, g_2)$:

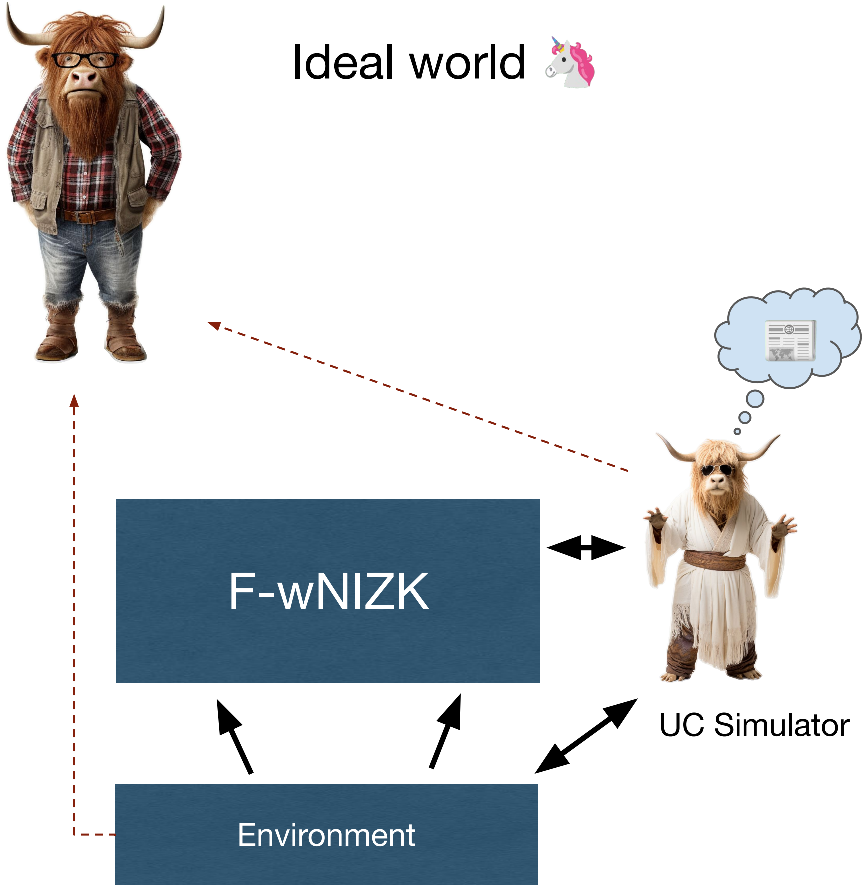
$\quad Obs \leftarrow Obs \,||\, (g_1, g_2)$

$\quad \text{return } \tau(\tau^{-1}(g_1) + \tau^{-1}(g_2))$

👏 **Good**: UC-simulator can extract π using *Obs*!

🚨 **Issue**: No ZK

⚔️ **Attack**: Check *Obs* to see whether π was honestly computed or via simulation trapdoor



Design challenges

👷 Requirements

- 👁️ Simulator **must see** group operations made by environment.
    - Required for extraction.
- 🙈 Environment **must not see** group operations made by the simulator.
    - Required for ZK.

- 💡 Solution: Partial observability via **domain separation**.

Design challenges

👷 Requirements

relevant

- 👁️ Simulator **must see** group operations made by environment.
  - Required for extraction.
- 🙈 Environment **must not see** group operations made by the simulator.
  - Required for ZK.

- 💡 Solution: Partial observability via **domain separation**.

<u>Observation rules</u> (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

Restricted observable global generic groups in UC

(simplified)

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

`public` generators $h_s \in S$ for sessions s

`public` observation list *Obs*

op(g_1, g_2):

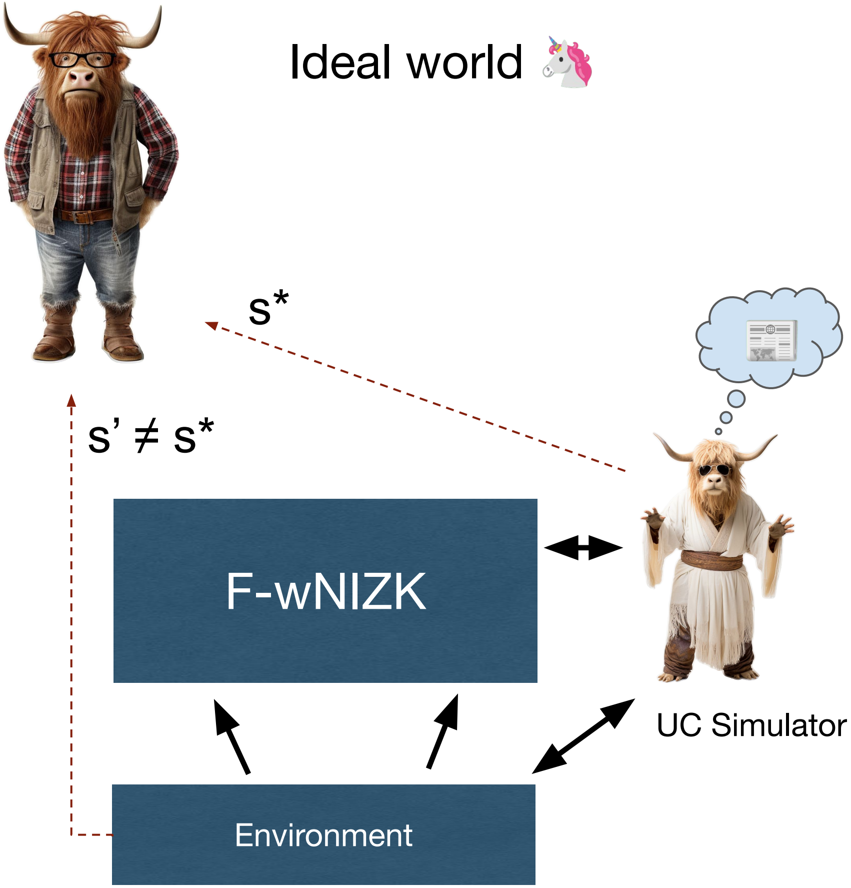
  if g_1 or g_2 contains some *foreign* h_s **then**

    $Obs \leftarrow Obs \,||\, (g_1, g_2)$

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

👏 UC-simulator **can simulate** unobserved.

👏 UC-simulator **can extract** π using *Obs*.



The actual G-oGG model in the paper

- 📚 Bookkeeping with polynomials.
    - (“does g contain some foreign h_s?”)
- 👥 Multiple generators per session
- 🤷‍♂️ Oblivious sampling
- 💫 Pairing groups

Our result

Theorem 1. *Π-$\mathtt{G16}$ UC-realizes $\mathcal{F}$-$\mathtt{wNIZK}$ in the $\mathcal{F}$-$\mathtt{CRS}$-hybrid model in the presence of $\mathcal{G}$-$\mathtt{oGG}$. Concretely, for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}_{\mathtt{G16}}$ such that for every $\mathcal{Z}$ that makes at most $q_{\mathcal{Z}}$ queries to $\mathcal{G}$-$\mathtt{oGG}$, $q_{\mathcal{P}}$ queries to the* Prove *interface, and $q_{\mathcal{V}}$ queries to the* Verify *interface,*

$$
\begin{aligned}
&|\Pr[\mathsf{EXEC}_{\mathcal{F}\text{-}\mathtt{wNIZK},\mathcal{Z},\mathcal{S}_{\mathtt{G16}},\mathcal{G}\text{-}\mathtt{oGG}}(\lambda, z) = 1] - \Pr[\mathsf{EXEC}_{\Pi\text{-}\mathtt{G16},\mathcal{Z},\mathcal{A},\mathcal{G}\text{-}\mathtt{oGG}}(\lambda, z) = 1]| \\
&\leq 72 \cdot d \cdot (m + d + q_{\mathcal{Z}} + (m + d)q_{\mathcal{P}} + \ell q_{\mathcal{V}} + 1)^2/(p - 1)
\end{aligned}
$$

and $\mathcal{S}_{\mathtt{G16}}$ performs in total the following operations:

- *at most $3q_{\mathcal{P}} + 9q_{\mathcal{V}} + 2q_{\mathcal{Z}} + 3d + m + 8$ queries to $\mathcal{G}$-$\mathtt{oGG}$*
- *at most $(2\ell + 8)q_{\mathcal{P}} + (3q_{\mathcal{Z}} + 2\ell + 2)q_{\mathcal{V}} + (d + 1)(3m + 11)$ field operations*

  where d, m, ℓ depend on the circuit size (see Functionality 6).

Groth16 UC proof challenges

Extraction

🏰 **Strategy:**
Observe group operations to extract coefficients = *w*

⚠️ **Challenge:**
Can only observe *my* session's generators

💡 **Solution:**
Argue: Foreign generators don't mess up extraction.

Simulation

🏰 **Strategy:**
Use CRS trapdoor to create proof without *w*

⚠️ **Challenge:**
Simulator's operations must not be observable

💡 **Solution:**
Easy: Simulator legally operates on CRS elements

Consequences

- 🥳 **Groth16** (as-is) **is UC-secure**
  - No need for compilers.

- 🔮 Our model **also enables** KZG/IOPs, Schnorr, … in UC

- 🚫 **Restrictions**:
  - GGM → **No circuit** for group operations
  - Some protocols **cannot deal** with (partial) observability. Workaround in paper.