On the Memory-Tightness of Hashed ElGamal

Ashrujit Ghoshal
University of Washington

Stefano Tessaro
University of Washington

Eurocrypt 2020

Security reductions

assumption $\Rightarrow$ scheme

P $\Rightarrow$ **S**

CDH, DDH, DL, factoring …

ElGamal, Cramer-Shoup, ECDSA, RSA-OAEP ⋯

Reduction R

$B = R^A$ $\Leftarrow$ A

Security reductions

$B = R^A$

time t_B

advantage ε_B

Reduction R

A

time t_A

advantage ε_A

Tight reductions

$B = R^A$

Reduction R

$\Longleftarrow$

A

time t_B

advantage ε_B

time t_A

advantage ε_A

Goal: tightness $\Longrightarrow\ t_B \approx t_A, \varepsilon_B \approx \varepsilon_A$

Time is not the only important resource!

Security reductions: memory perspective [ACFK17]

$B = R^A$

Reduction R

$\Leftarrow$

A

time t_B

memory m_B

advantage ε_B

time t_A

memory m_A

advantage ε_A

Memory-tight reductions [ACFK17]

$B = R^A$

uses memory m_R

Reduction R

⇦

A

memory m_B

memory m_A

$$m_B = m_A + m_R$$

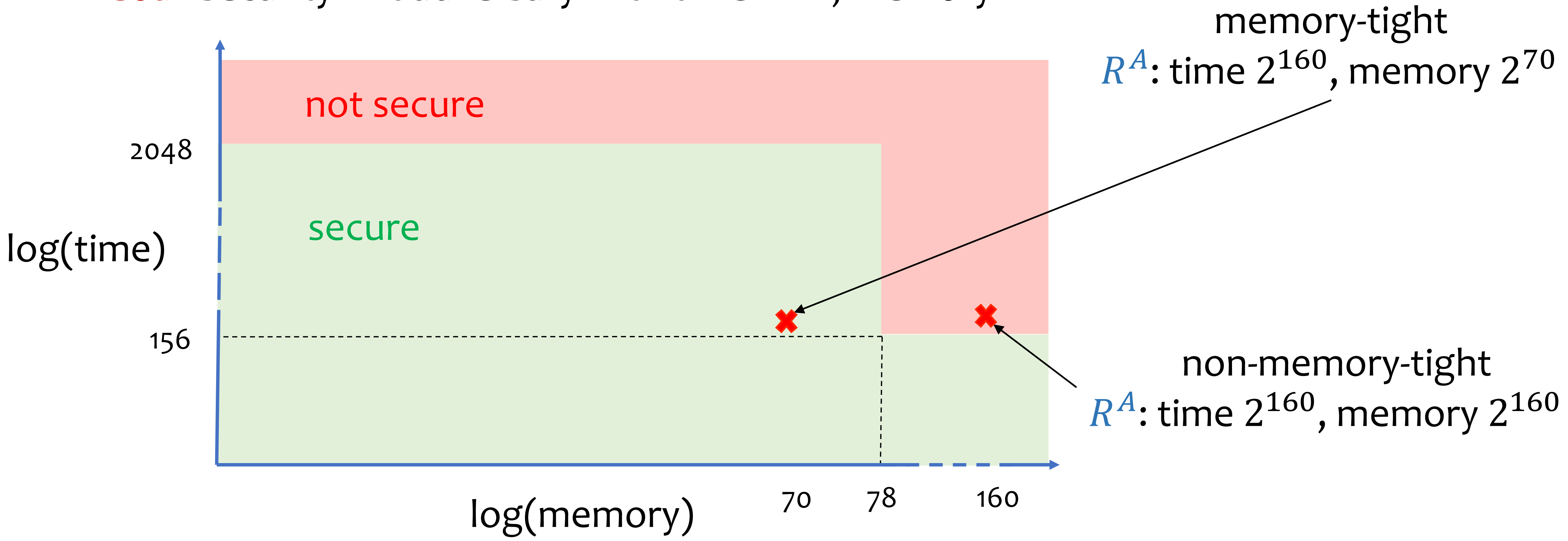
Goal: memory-tightness $\Longrightarrow\ m_B \approx m_A$

Common proof technique: m_R small ⇒ memory-tight reduction

Motivation: more memory ⟹ faster solution

Discrete logarithm (DL) in prime fields

Goal: security wrt adversary with time 2^{160}, memory 2^{70}



Can we always make a reduction memory-tight?

This talk: certain reductions **cannot** be memory-tight, **provably**

Prior work

- mUFCMA to UFCMA [ACFK17]
- mCR_t to CR_t [ACFK17,WMHT18]
- mU-mOW to mU-OW [WMHT18]

generic

Here

Hashed ElGamal

concrete scheme

Hashed ElGamal used in practice eg. SECG SEC-1, ISO/IEC 18033-2, IEEE 1363a and ANSI X9.63

Hashed ElGamal KEM

Group $\mathbb{G}$, generator g, order p

Gen

$pk \leftarrow g^{sk}, sk$

pk

Encap

$u \overset{\$}{\leftarrow} \mathbb{Z}_p$

$C \leftarrow g^u, K \leftarrow H(pk^u)$

(sk, C)

Decap

$K \leftarrow H(C^{sk})$

KEM-CCA security ≡ **Oracle Diffie-Hellman assumption** [ABR `01]

Oracle Diffie-Hellman assumption (ODH)

$u, v \xleftarrow{\$} \mathbb{Z}_p$ 

$K_0 \leftarrow H(g^{uv}), K_1 \xleftarrow{\$} \{0,1\}^{hLen}$ 

$b \xleftarrow{\$} \{0,1\}$

g^u, g^v, K_b

$D_v(Y)$

D_v

$$D_v(Y) = \begin{cases} H(Y^v) & \text{if } Y \neq g^u \\ \perp & \text{otherwise} \end{cases}$$

b'

$$\Pr[b = b'] = \frac{1}{2} + \text{negl}$$

ODH in the random oracle model

$u, v \xleftarrow{\$} \mathbb{Z}_p$

$K_0 \leftarrow H(g^{uv}), K_1 \xleftarrow{\$} \{0,1\}^{hLen}$

$b \xleftarrow{\$} \{0,1\}$

g^u, g^v, K_b

$D_v(Y)$

$\boldsymbol{D_v}$

$$D_v(Y) = \begin{cases} H(Y^v) & \text{if } Y \neq g^u \\ \perp & \text{otherwise} \end{cases}$$

$H(X)$

$\boldsymbol{H}$

random oracle

b'

$$\Pr[b = b'] = \frac{1}{2} + \text{negl}$$

SDH $\Longrightarrow$ ODH [ABR '01]

Strong Diffie-Hellman assumption (SDH) (aka gap-DH)

$u, v \xleftarrow{\$} \mathbb{Z}_p$ 

g^u, g^v

$O_v(X, Y)$

$\boldsymbol{O_v}$

Z

$$O_v(X,Y) = \begin{cases} 1 & \text{if } Y = X^v \\ 0 & \text{otherwise} \end{cases}$$

$\Pr[Z = g^{uv}] = \text{negl}$

Strong Diffie-Hellman (SDH) $\Longrightarrow$ ODH [ABR '01]

Theorem. ODH-adversary using memory $m_A \Longrightarrow$ **SDH**-adversary using memory m_B

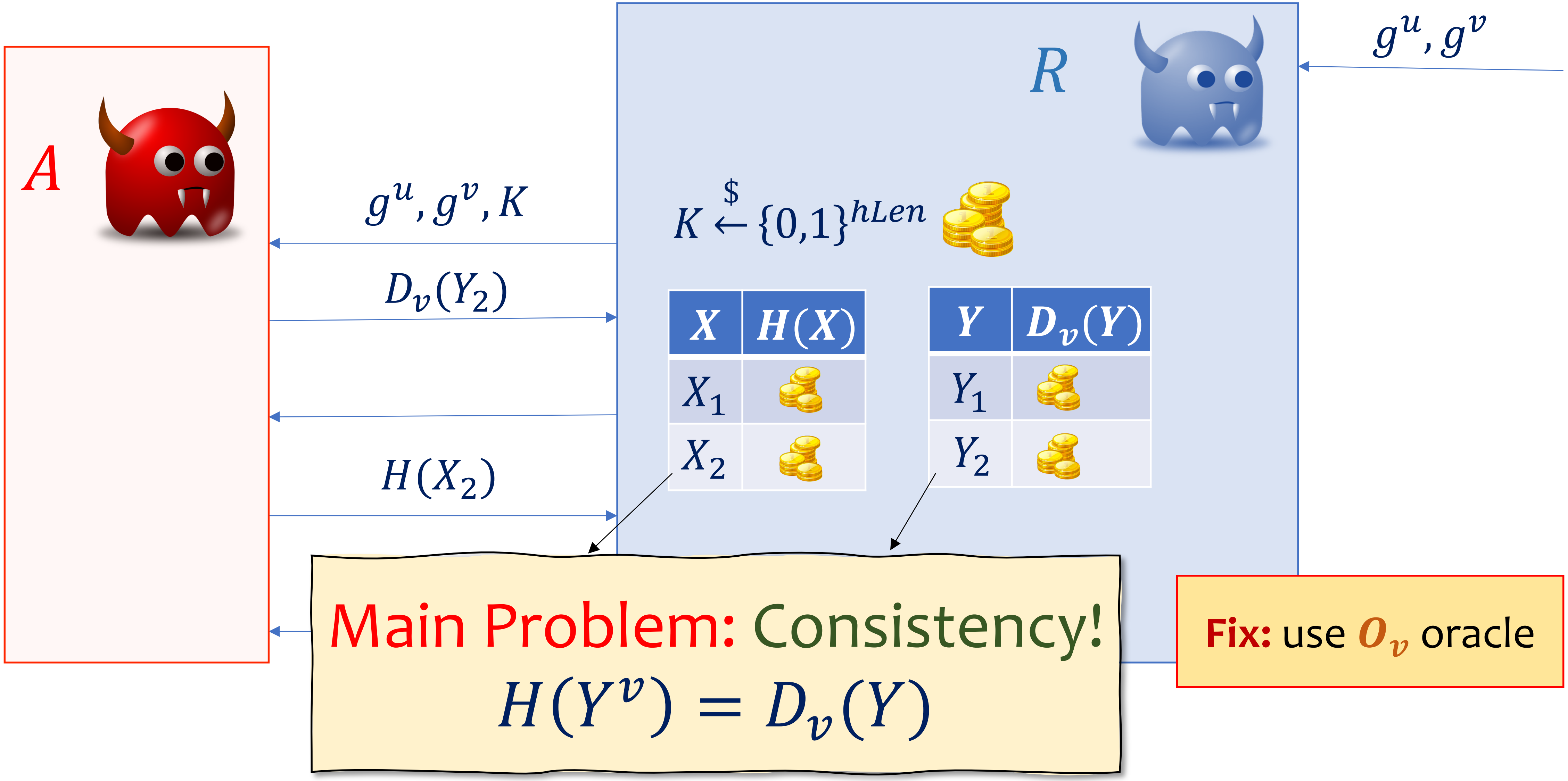
$$m_B = m_A + O(q_H + q_D)$$

H queries

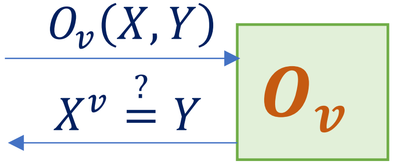
D_v queries

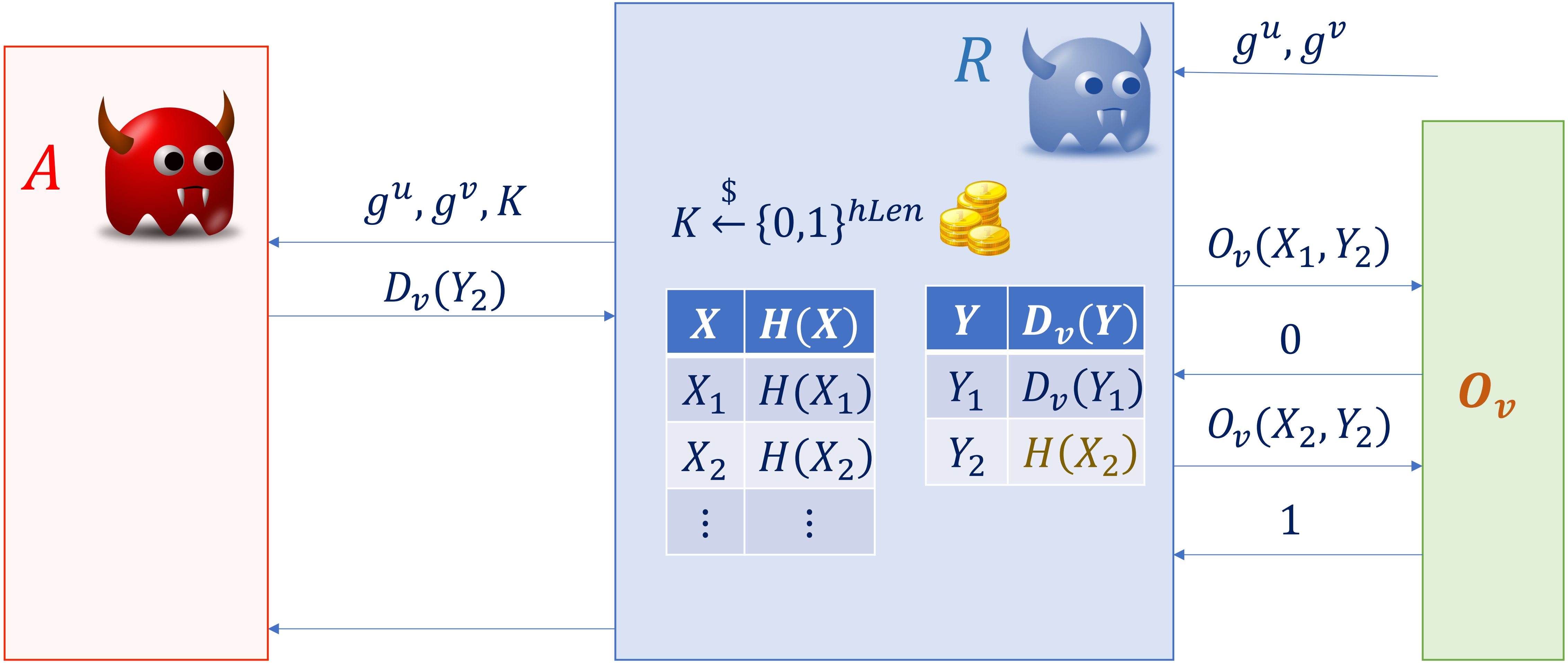
not memory-tight!

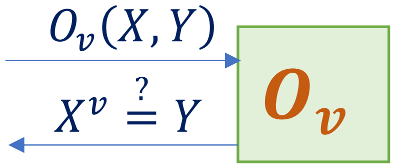
SDH⇒ODH: the reduction



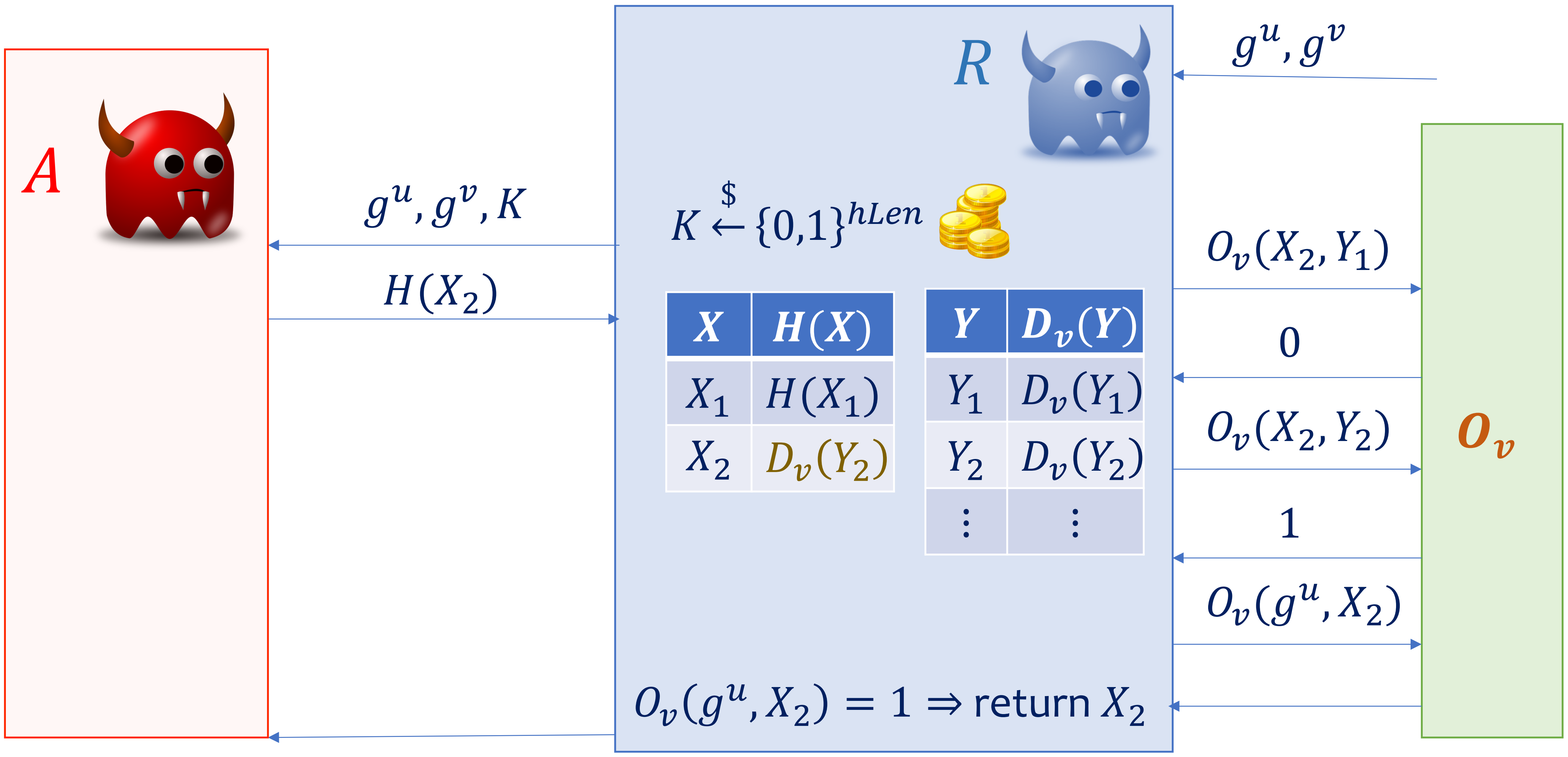
SDH⇒ODH: the reduction- D_v queries







SDH⇒ODH: the reduction- H queries



X	$H(X)$
X_1	$H(X_1)$
X_2	$D_v(Y_2)$

Y	$D_v(Y)$
Y_1	$D_v(Y_1)$
Y_2	$D_v(Y_2)$
⋮	⋮

Main theorem

inefficient

Theorem. $\forall k\ \exists O(k)$-query ODH-adv A^* s.t.

- $\mathrm{Adv}_{\mathbb{G}}^{\mathrm{ODH}}(A^*) \approx 1$,
- $\forall$ PPT black-box reductions R using memory m,

$$\mathrm{Adv}_{\mathbb{G}}^{\mathrm{SDH}}\left(R^{A^*}\right) = \text{non-negl} \Rightarrow m = \Omega(k \log p).$$

Issue: For which groups $\mathbb{G}$? DL easy in $\mathbb{G}$ $\Rightarrow$ memory tight R

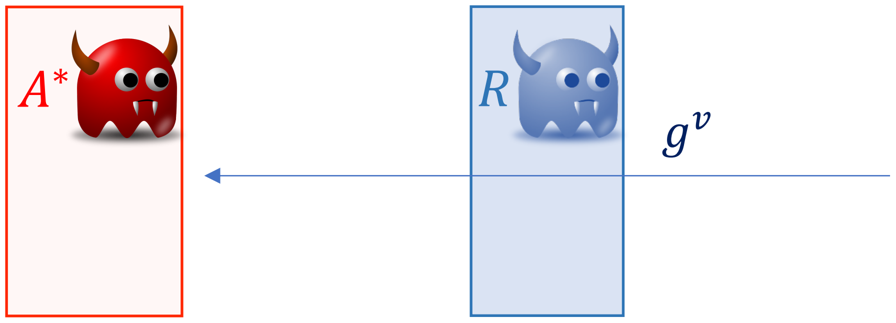
Resolution: R only makes black-box access to the group $\Rightarrow$ **generic group model**

Main theorem

Theorem. In the generic group model, $\forall k\ \exists O(k)$-query ODH-adv A^* s.t.

- $\mathrm{Adv}^{\mathrm{ODH}}(A^*) \approx 1$,
- $\forall$ PPT black-box reductions R using memory m,

$$\mathrm{Adv}^{\mathrm{SDH}}\left(R^{A^*}\right) = \text{non-negl} \Rightarrow m = \Omega(k \log p)\,.$$



forwarding

+ no rewinding!

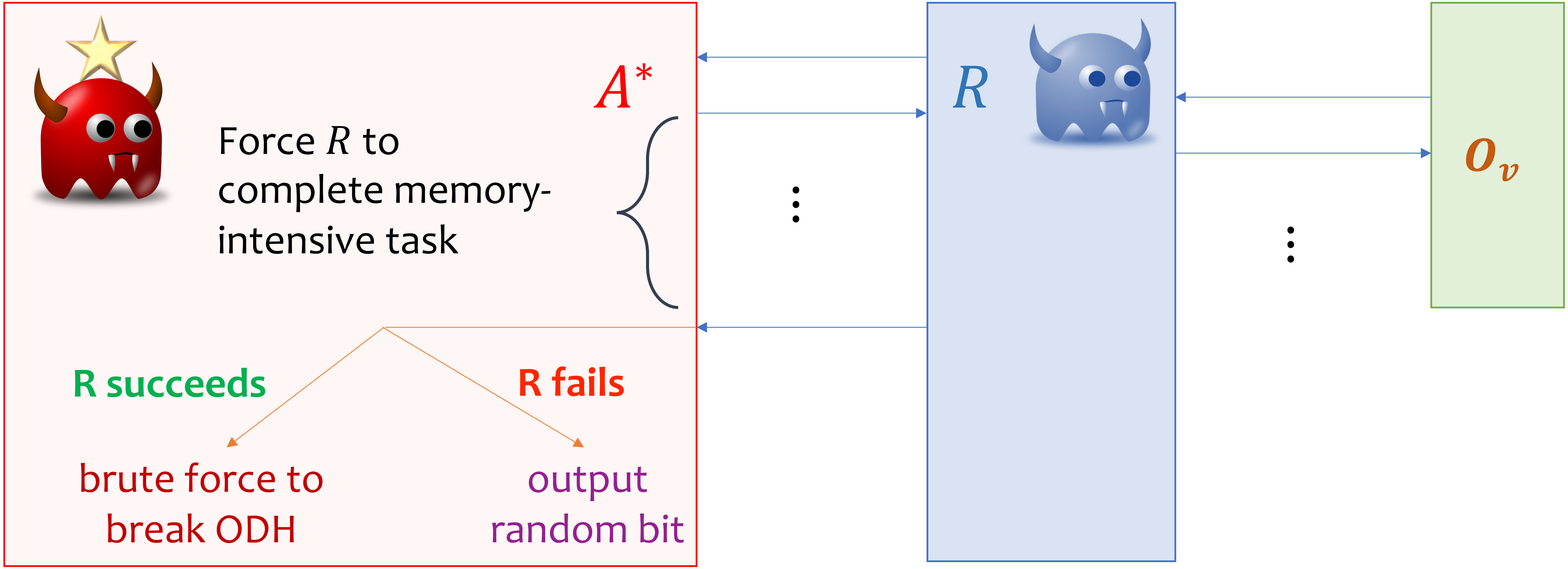
Main theorem

Theorem. In the generic group model, $\forall k\ \exists O(k)$-query ODH-adv A^* s.t.

- $\mathrm{Adv}^{\mathrm{ODH}}(A^*) \approx 1$,
- $\forall$ PPT restricted black-box reductions R using memory m,

$$\mathrm{Adv}^{\mathrm{SDH}}(R^{A^*}) = \mathrm{non{-}negl} \Rightarrow m = \Omega(k \log p) .$$

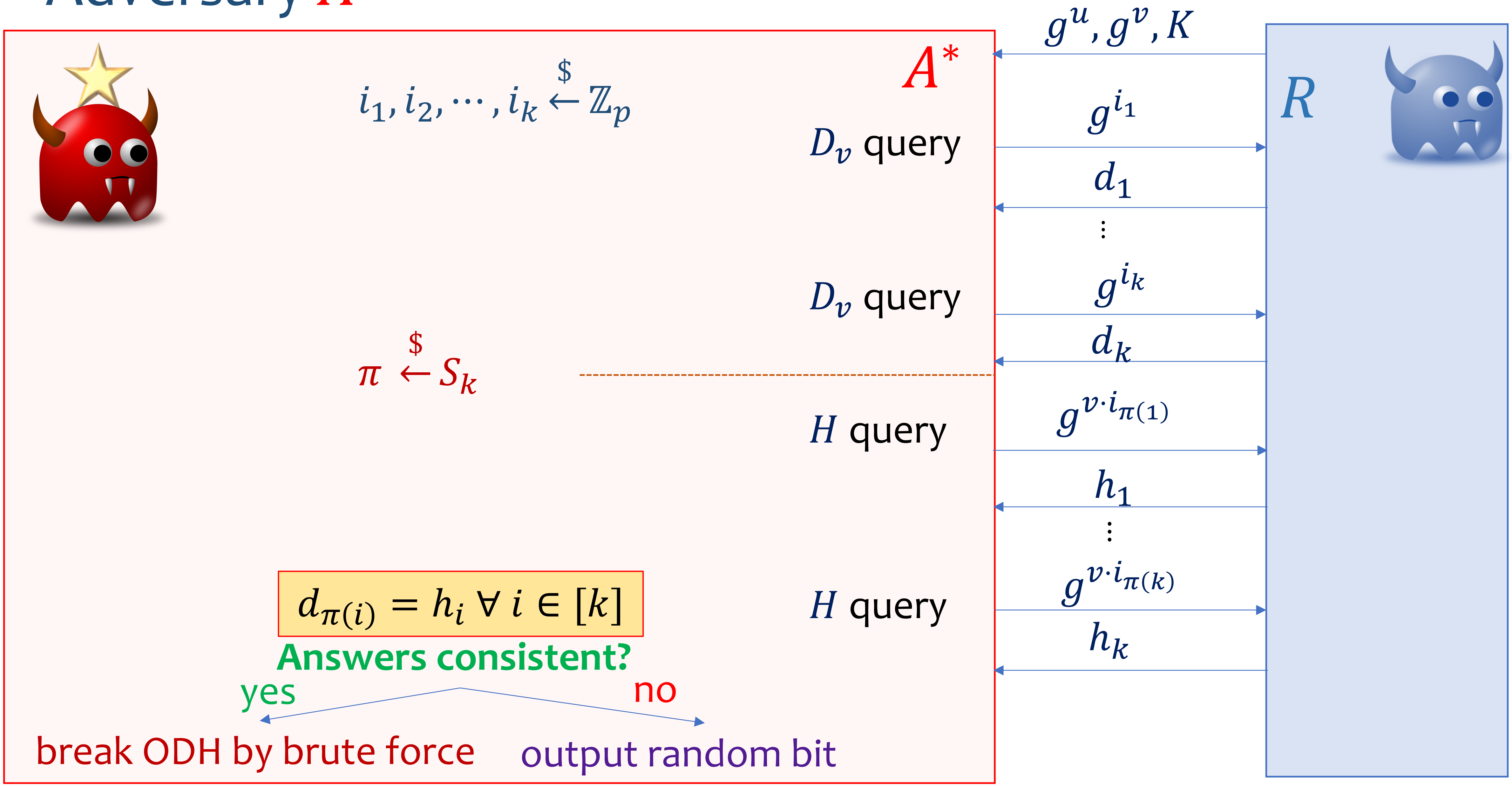
Constructing A^*



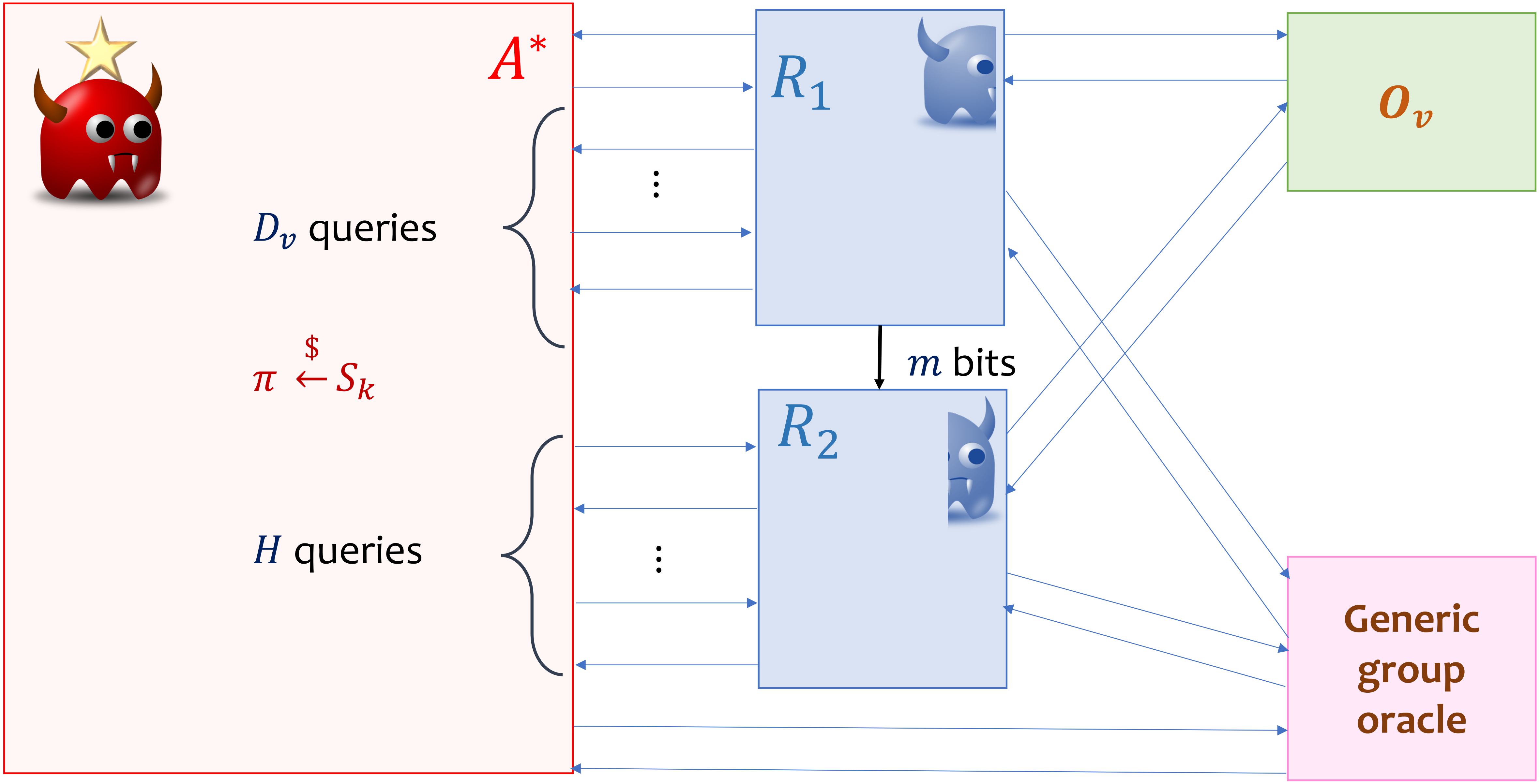
Intuition: A^* is useful to R only if R accomplishes memory-intensive task

Recall: $D_v(Y) = H(Y^v)$

Adversary A^*

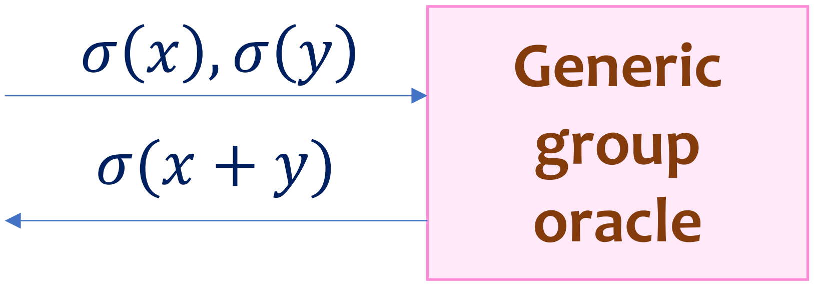


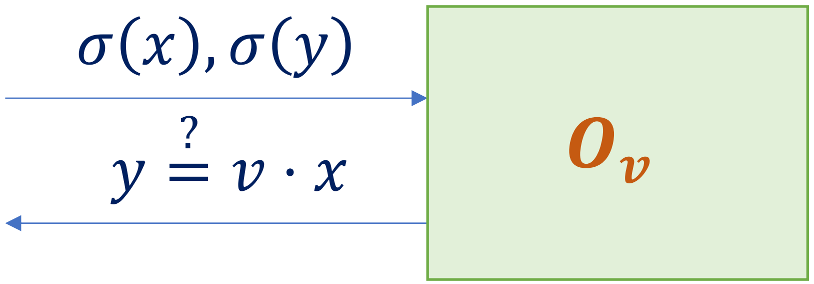
Proof setting



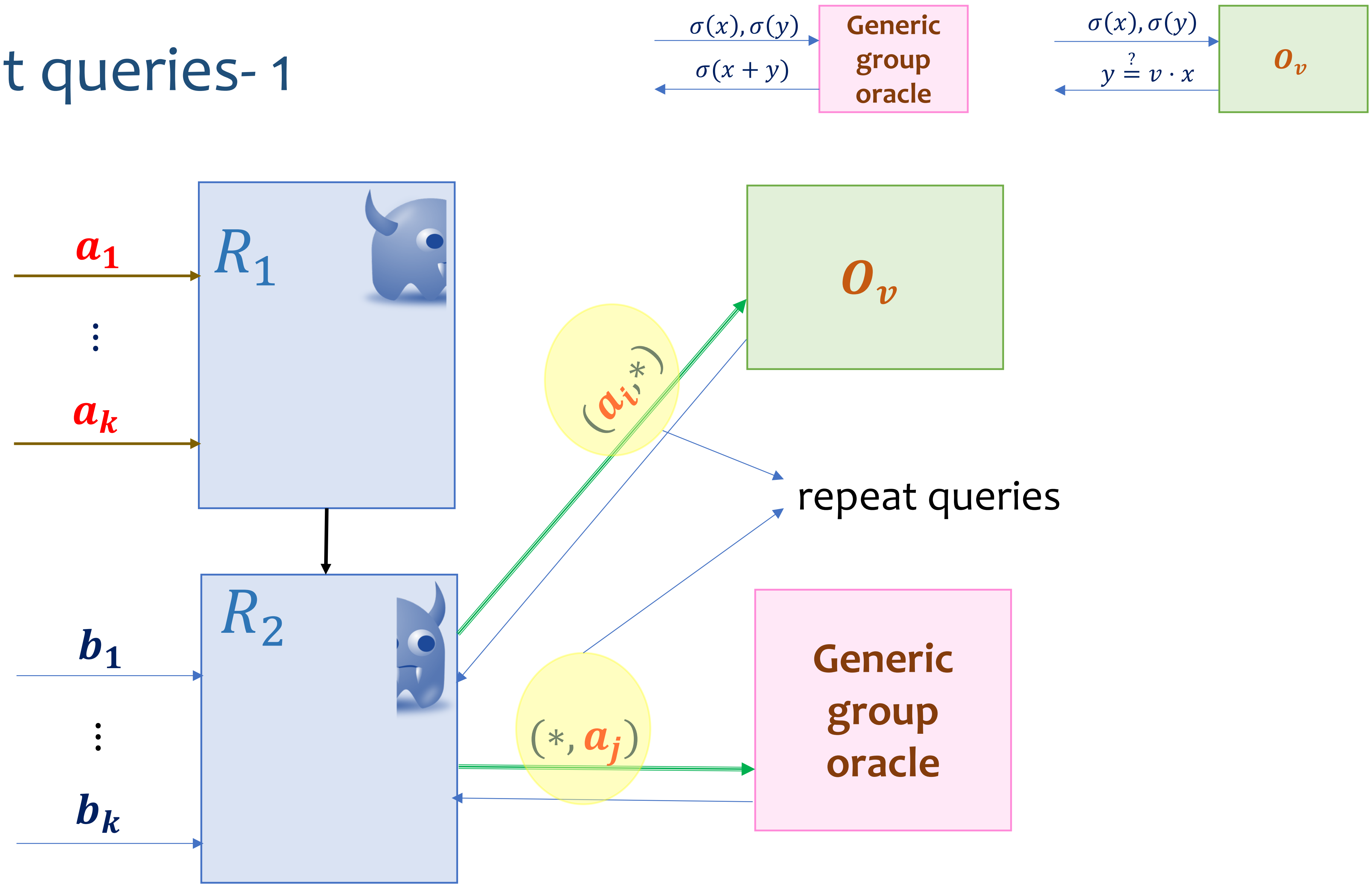
Generic group model [Shoup 97, Maurer 05]

$$\sigma: \mathbb{Z}_p \to \{0,1\}^\lambda$$
$$x \in \mathbb{Z}_p:\ \sigma(x) \triangleq g^x$$

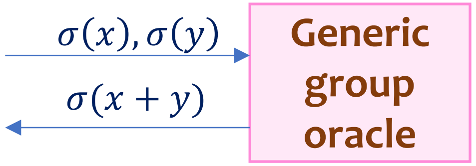


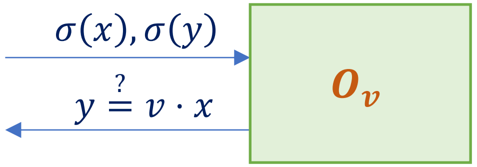


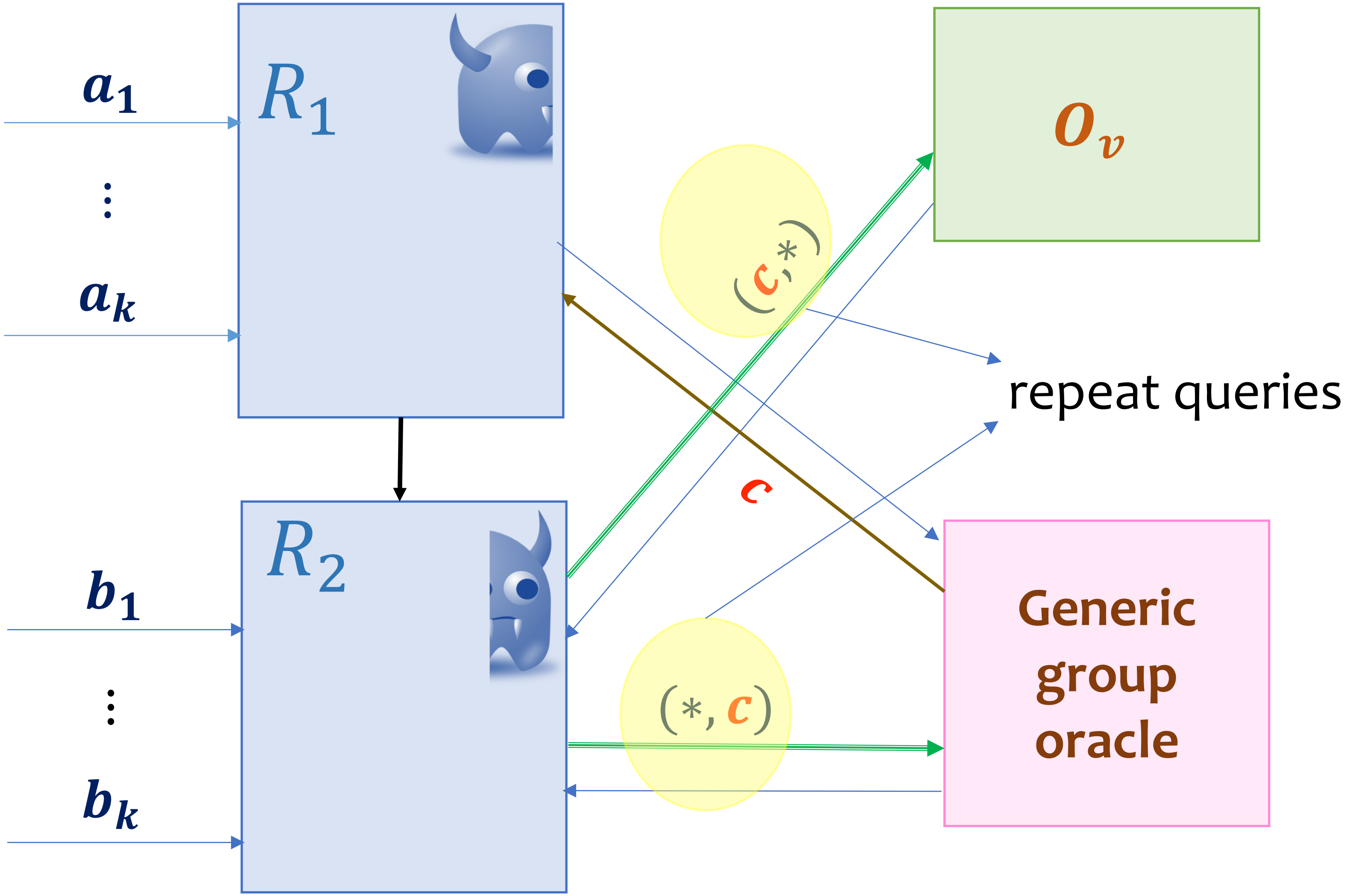
Repeat queries- 1



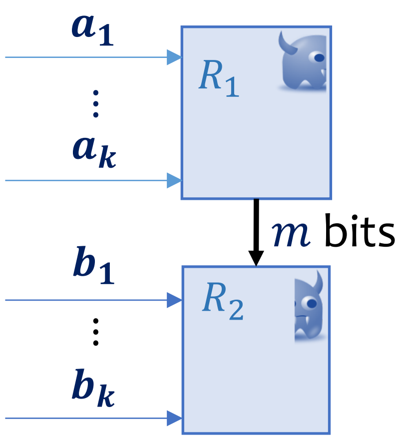
Repeat queries- 2







Proof overview



(R_1, R_2) answer consistently

Many $\left(> \frac{k}{80}\right)$ repeat queries

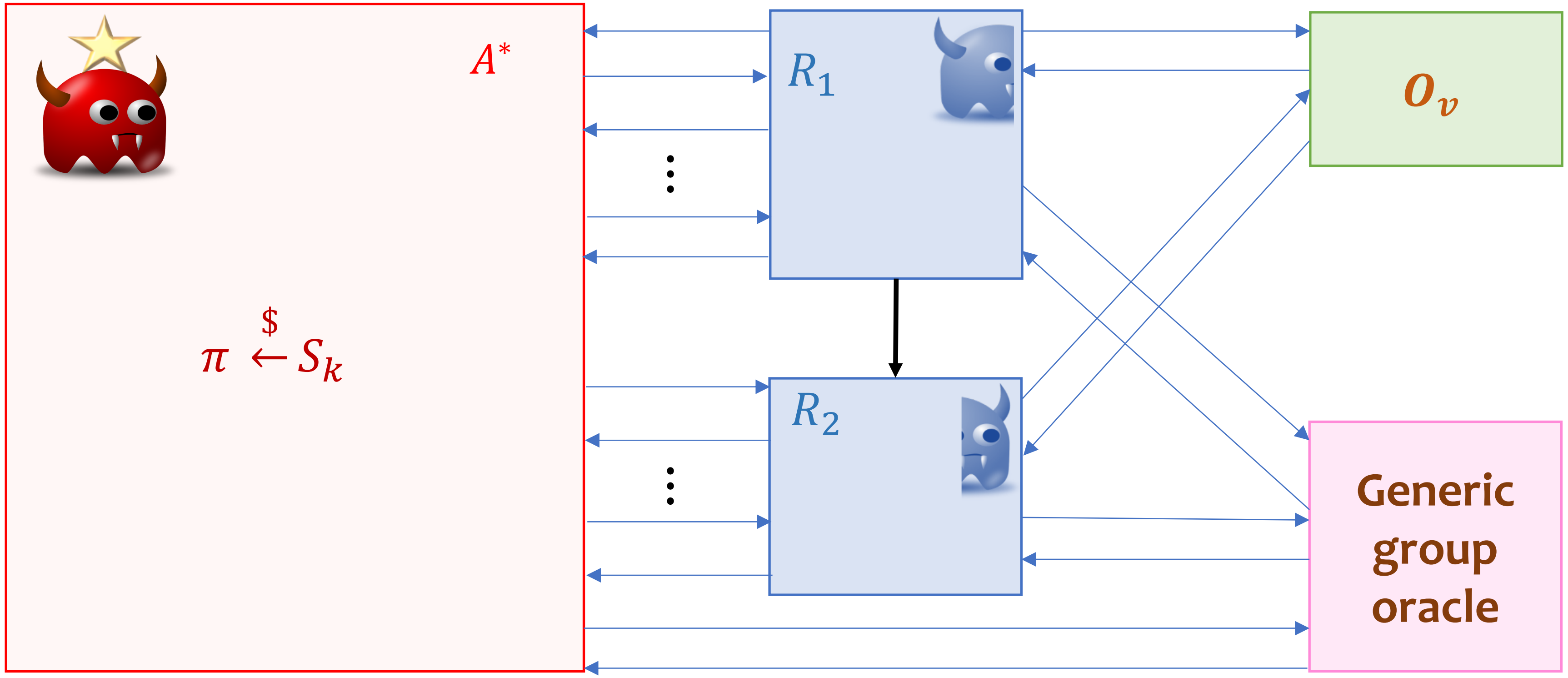
Need $\boldsymbol{m = \Omega(k \log p)}$: intuitive, proof by compression argument, many subtleties

Few $\left(\leq \frac{k}{80}\right)$ repeat queries

Winning adversary against the **permutation game**

Advantage negligible

The reduction's perspective

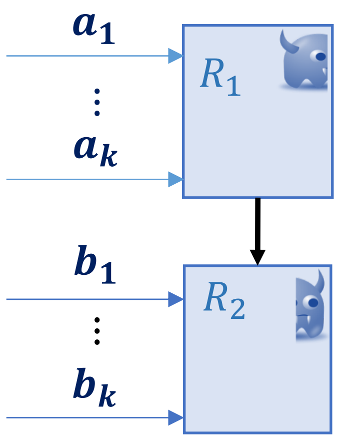


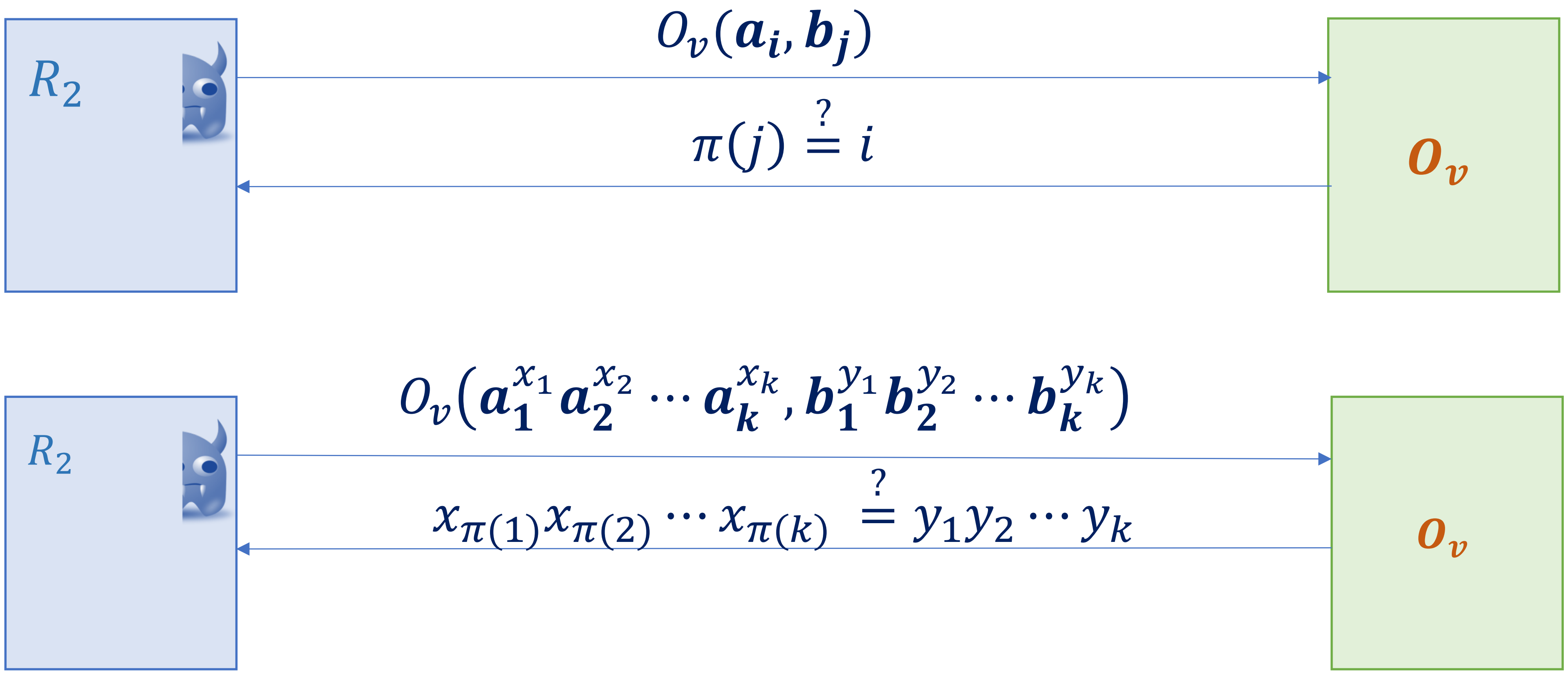
R_2 needs to figure out π for consistent answers

$\rightarrow$Use O_v oracle!

Using the O_v oracle

$$a^v_{\pi(i)} = b_i$$





Permutation game captures exactly this setting, combinatorially

Permutation game (PG)

$$O(\vec{x}, \vec{y}) = \begin{cases} 1 & \text{if } x_{\pi(1)} x_{\pi(2)} \cdots x_{\pi(k)} = y_1 y_2 \cdots y_k \\ 0 & \text{otherwise.} \end{cases}$$

$\pi \xleftarrow{\$} S_k$

A

$O(\vec{x} \in \mathbb{Z}_p^k, \vec{y} \in \mathbb{Z}_p^k)$ →

←

π' →

O

$\mathrm{Adv}^{\mathrm{PG}}(A) = \Pr[\pi' = \pi]$

$\vec{x} = x_1 x_2 \cdots x_k$

$\vec{y} = y_1 y_2 \cdots y_k$

Lemma: If $(\overrightarrow{x_1}, \overrightarrow{y_1}), \cdots, (\overrightarrow{x_u}, \overrightarrow{y_u})$ are the queries by A that return 1 and $\mathrm{rank}(\overrightarrow{x_1}, \cdots, \overrightarrow{x_u}) \leq \frac{k}{80}$, then,

$$\mathrm{Adv}^{\mathrm{PG}}(A) = \mathrm{negl}\,.$$

(R_1, R_2) make few repeat queries ⇒ A of this form that wins PG if (R_1, R_2) answer consistently

Conclusions

- Impossibility result for a scheme with algebraic structure
- Impossibility result can be "bypassed"
  - Memory-tight reduction in the Algebraic Group Model [FKL18]

    Adv sends a representation of the group elements for every query
  - Concurrent work [Bhattacharya 20] complements our result

    Different Hashed ElGamal variant, pairings

Open problems

- Memory lower bound for rewinding R?

  Our conjecture: $m = \Omega(k \log k)$
- Separation for “memory-adaptive” reduction?
- Memory lower bound for concrete schemes without the generic group model?
- Memory lower bounds for other concrete schemes?


Thank
You