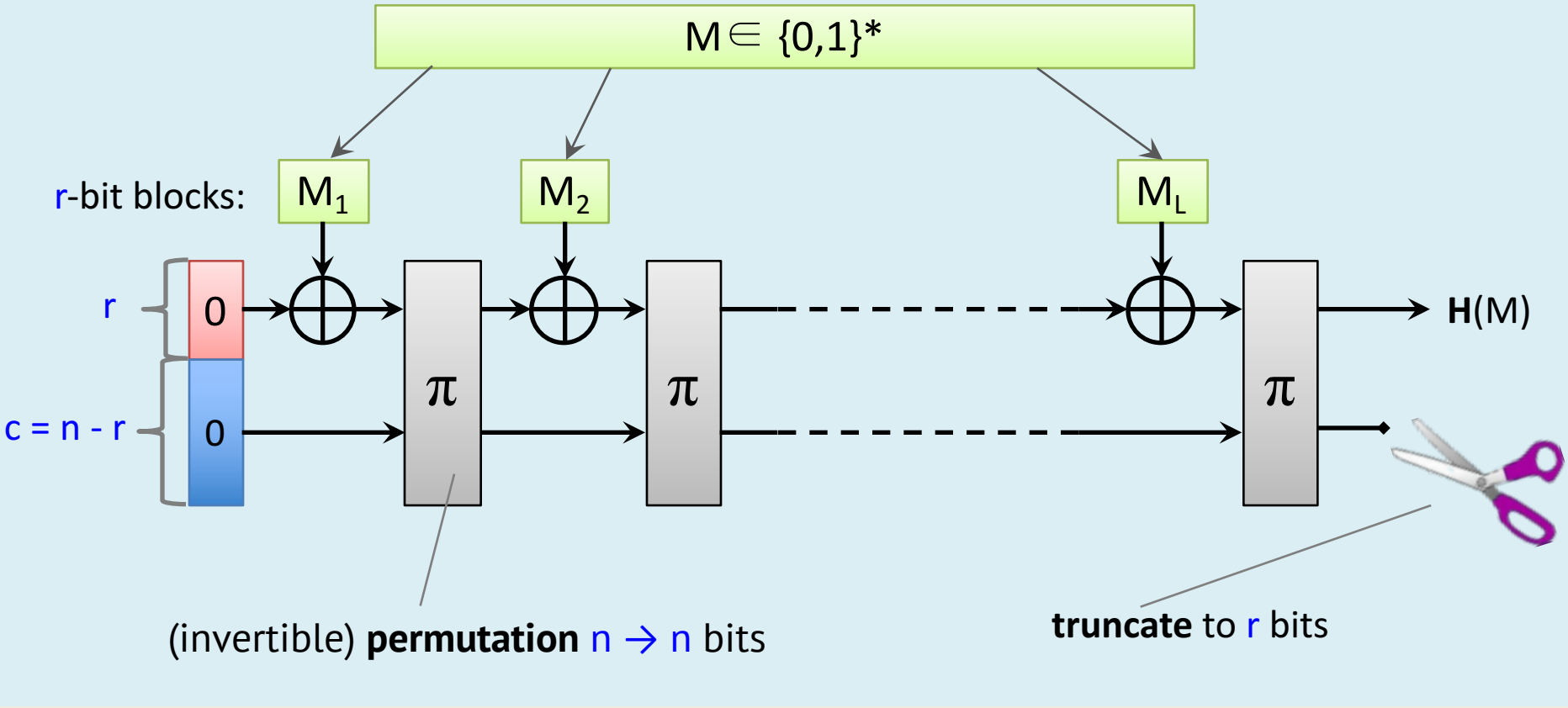
Sponge-based PRNGs
A Provable Security Perspective

Stefano Tessaro
UCSB

Base on joint work with **Peter Gaži** (IST Austria)

wr0ng
Paris, April 30, 2017

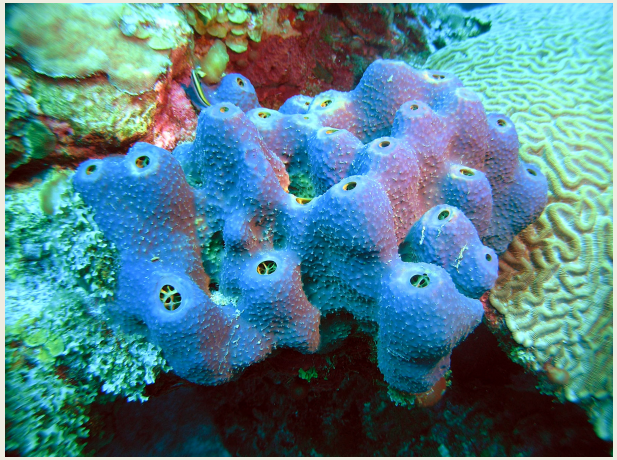
The Sponge Construction [BDPVA08]



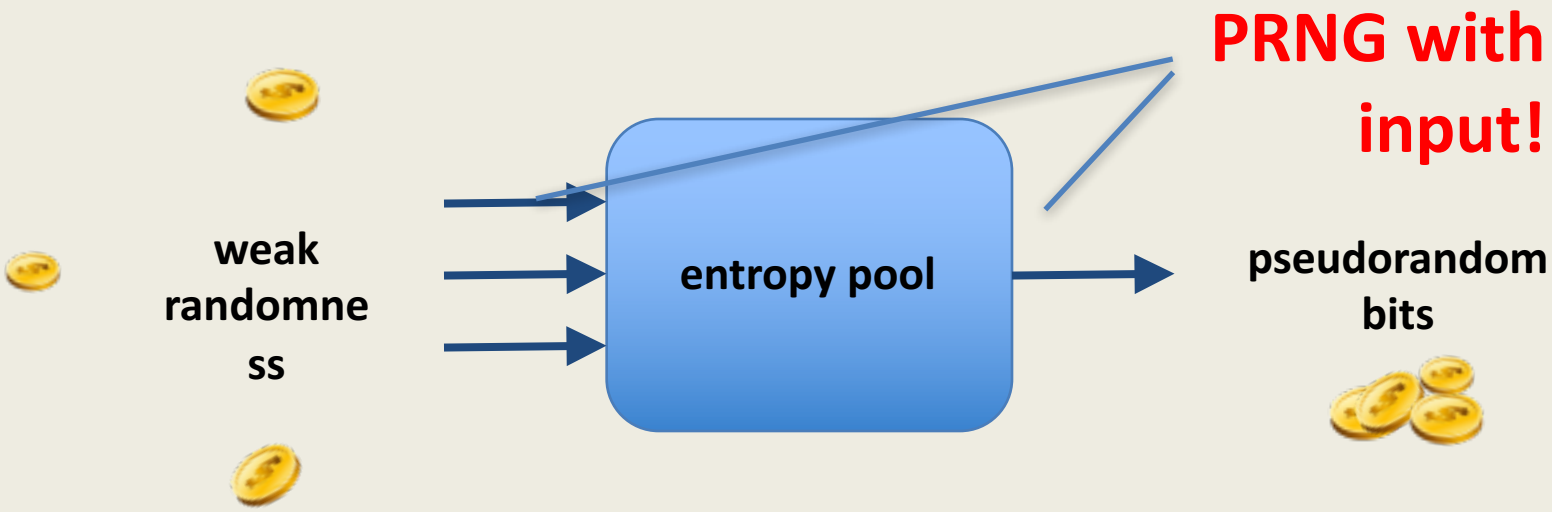
The Sponge Paradigm – Beyond hashing

The **sponge paradigm** has been used to build:

- Authenticated encryption schemes
- Message-authentication codes / PRFs
- PRNGs



Pseudorandom Number Generators



- **Few PRNGs come with security proofs.**
  [Barak-Halevi, CCS'15], [Dodis-Pointcheval-Ruhault-Vergnaud-Wichs, CCS'13], [Shrimpton-Tarashima, EC'15], [Dodis-Shamir-Stephens-Davidovitz-Wichs, C'15]

- **Real-world PRNGs rarely designed with provable security in mind!**

This talk, in a nutshell

Discuss state of the art on **sponge-based PRNGs**, and challenges in their provable security!

Talk based on: Peter Gaži and Stefano Tessaro. ***Provably Robust Sponge-Based PRNGs and KDFs.*** *EUROCRYPT '16*

Main take-home messages:

1. Sponge-based PRNGs are **elegant designs**.

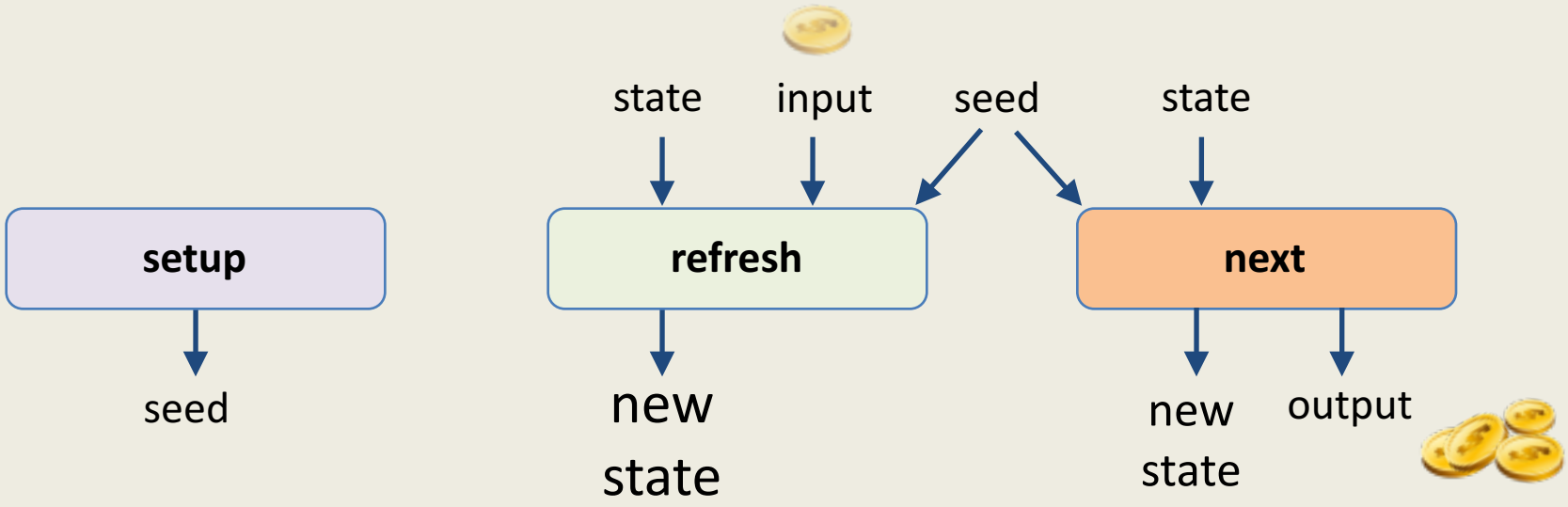
2. Proper analysis of sponge-based PRNGs presents several **technical challenges**.

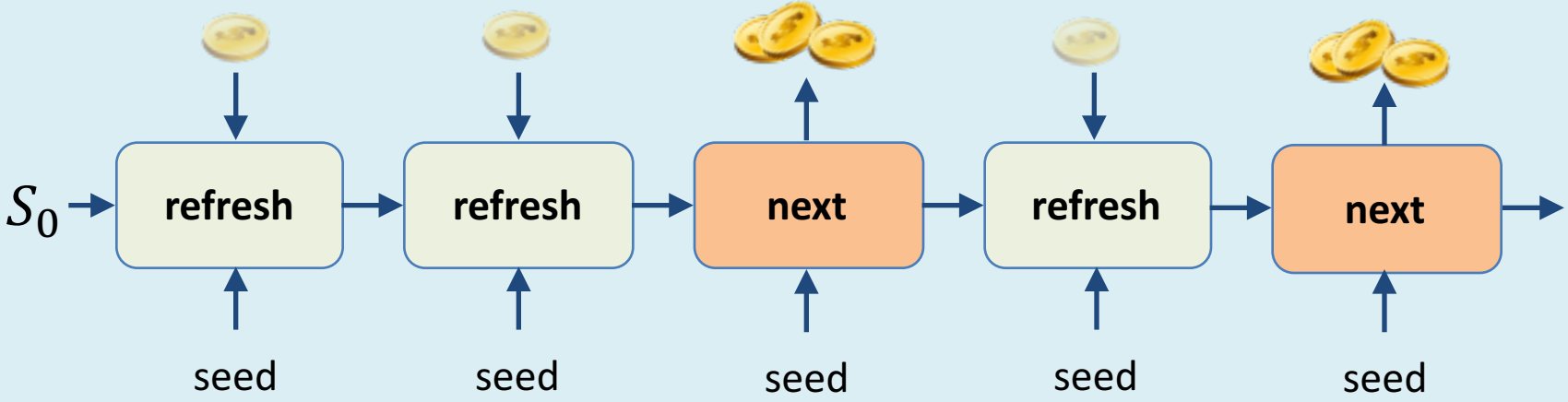
3. This will bring up some **food-for-thought**.

Roadmap of this talk

1. PRNGs: Sponge-based Instantiations
2. Provably-robust sponge-based PRNGs
3. Conclusions and open questions

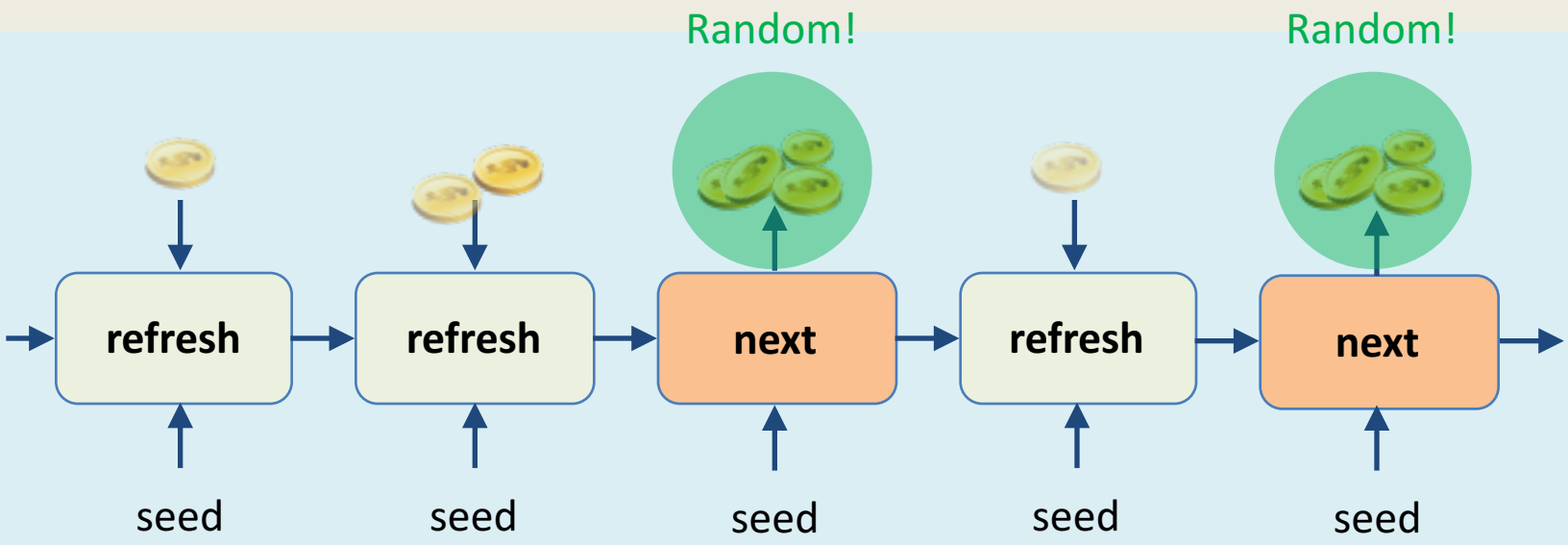
PRNGs with Input [DPRVW13]





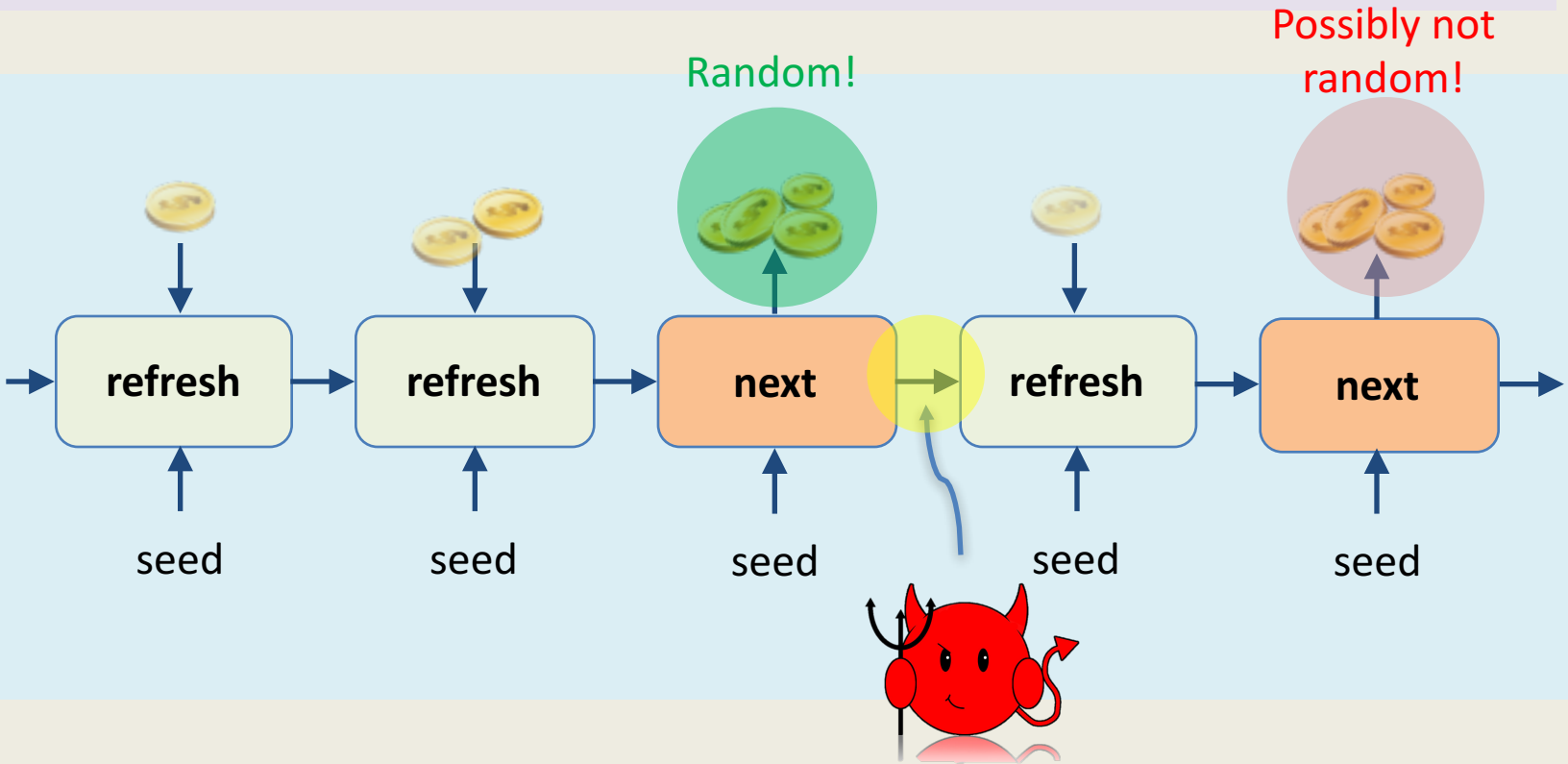
Desiderata – Pseudorandomness

Pseudorandomness: Output bits of **next** are indistinguishable from truly random bits, provided enough entropy is injected.



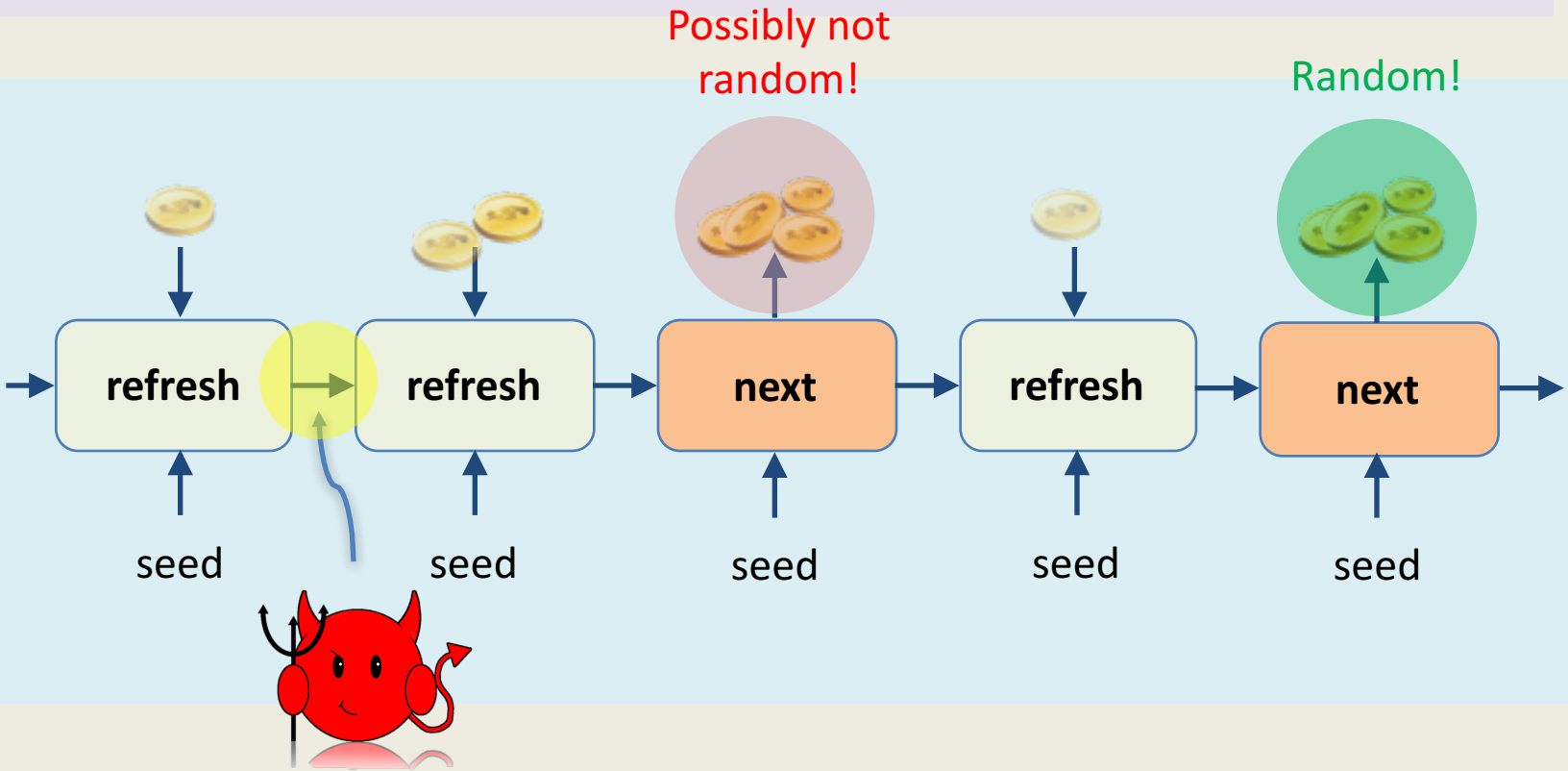
Desiderata – Forward secrecy

Forward secrecy: Even if the attacker compromises the state, it cannot distinguish previous outputs from random!

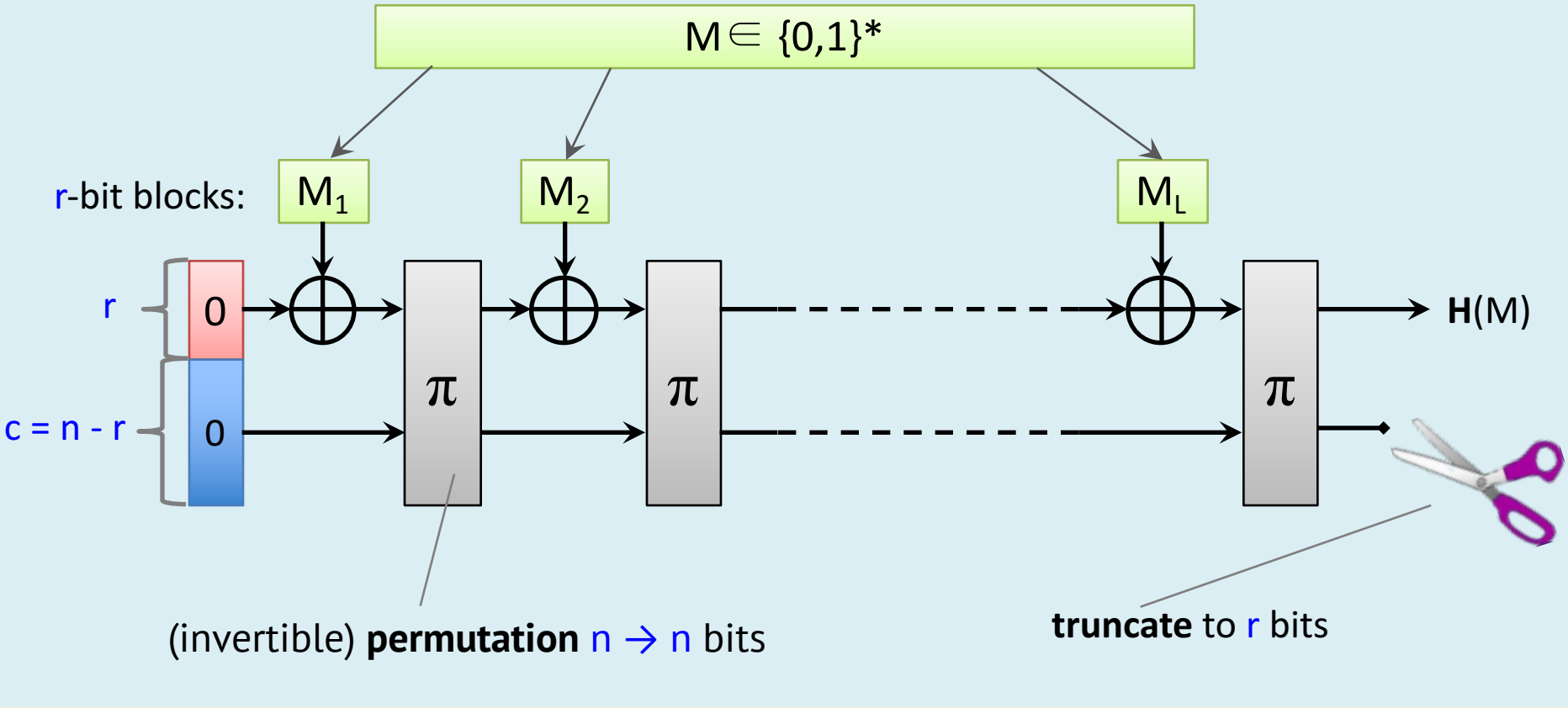


Desiderata – Backward secrecy

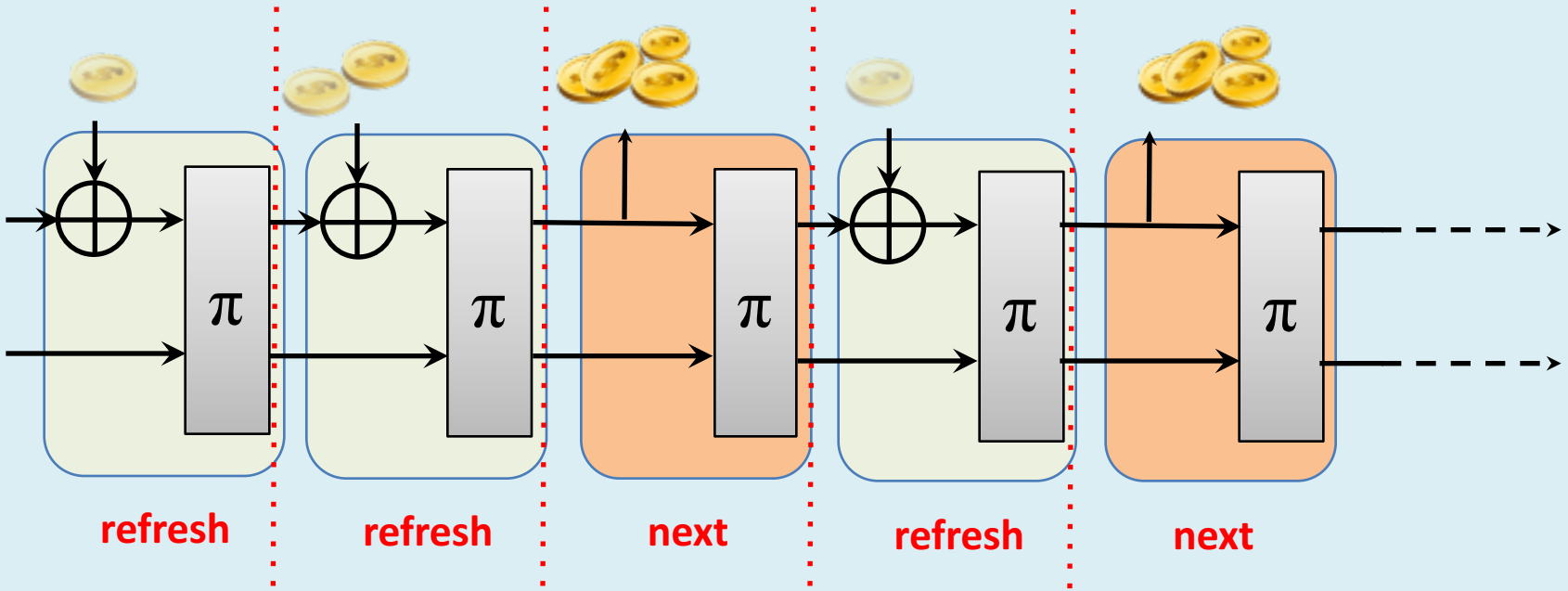
Backward secrecy: Even if the attacker compromises the state, future bits are pseudorandom after enough entropy is injected.



The Sponge Construction [BDPVA08]



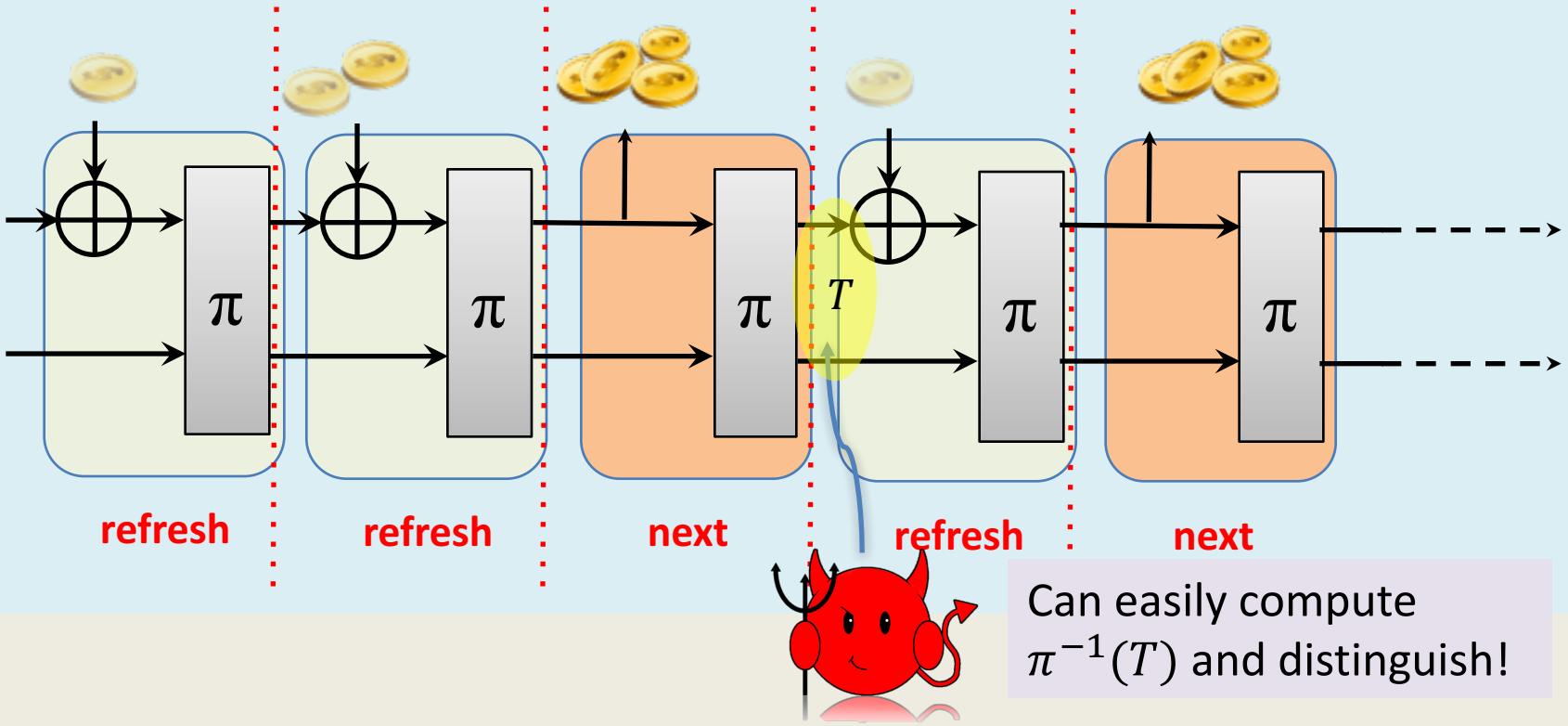
Sponge-based PRNGs: Existing Proposal [BDPvA10]



- simple and elegant
- analysis in simple model
- implemented, e.g., on microcontrollers [vHV14]

Three main issues with design + analysis we are aiming two resolve!

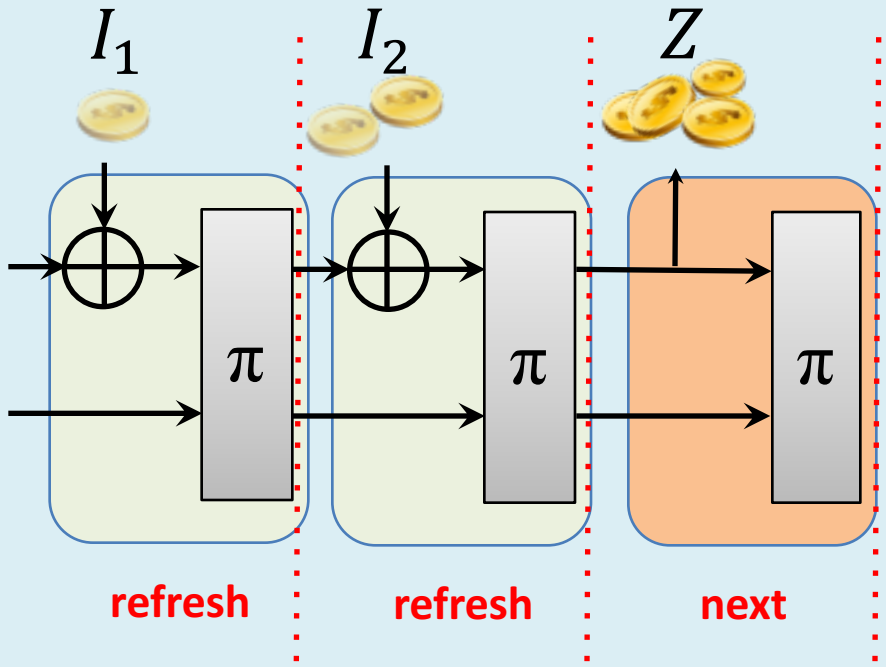
Problem 1: No Forward Secrecy



- recognized in [BDPVA10]
- proposed patch: zeroing upper bits after **next**
  - not analyzed

Problem 2: No Seed

Pseudorandomness: If inputs have sufficient entropy, then output should be uniform!



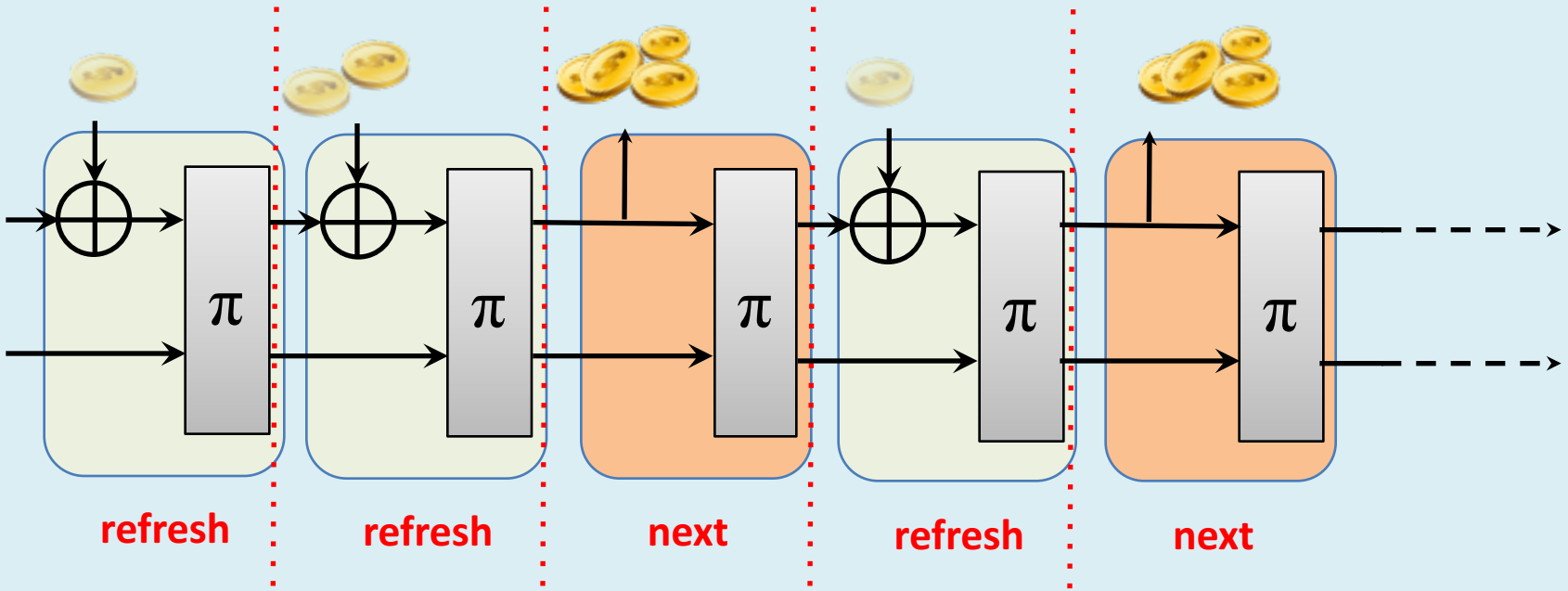
(I_1, I_2) uniformly distributed such that first bit of Z equals 0.

Clearly, Z is not pseudorandom!

Yet, (I_1, I_2) has almost max entropy! (only one bit loss)

[BDPVA10] did not have this issue, due to technical reasons in their proof … coming next …

Problem 3: Modeling the Permutation



Proofs for sponge-based construction rely on the random permutation model! I.e., π is random + adversary has access to π / π^{-1}

Previous attack: Input distribution depends on π!!!

Existing proofs: Distribution is independent of π!!!

Permutation-dependence and the seed: Why care?

Typical argument: Real-world distributions behave nicely!

Possible, but … it is not easy to characterize what “real-world distribution” means...

Roadmap of this talk

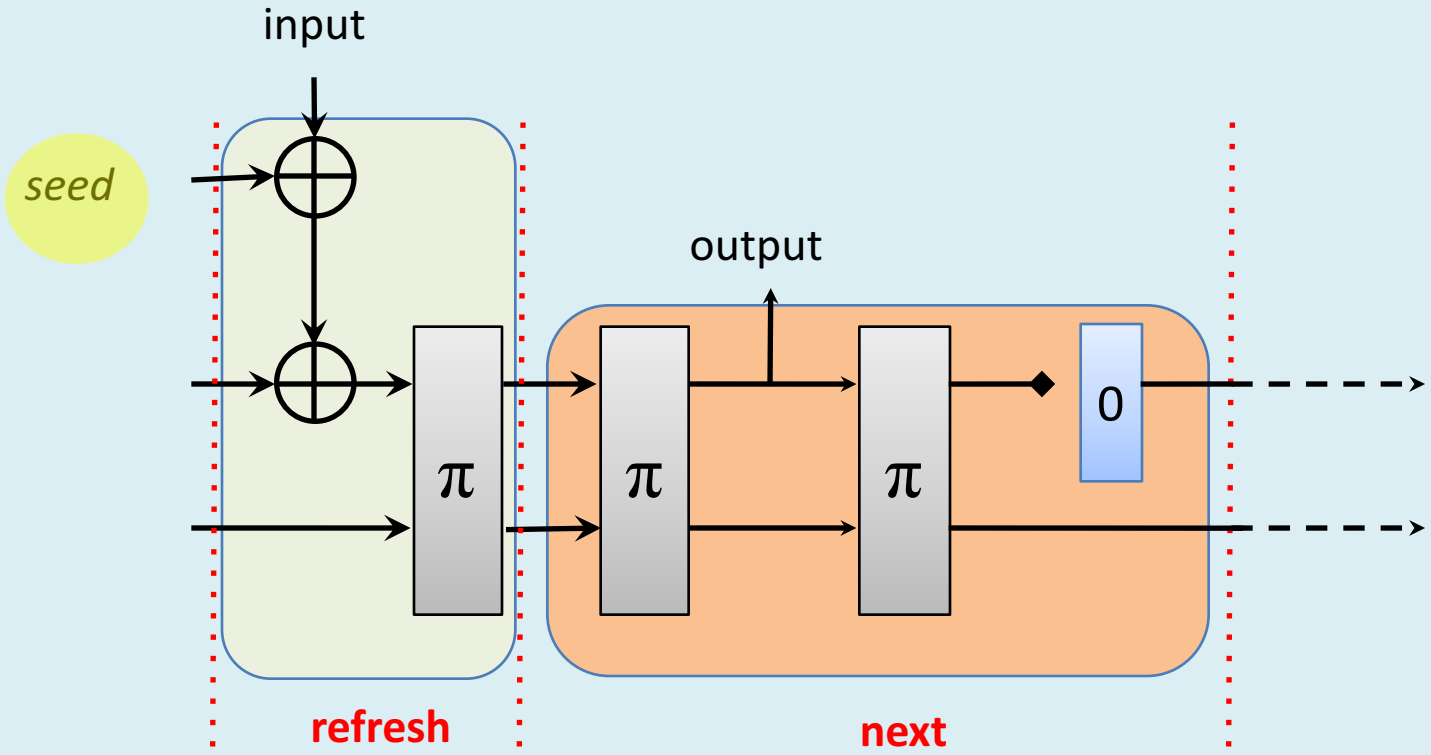
1. PRNGs: Sponge-based Instantiations
2. Provably-robust sponge-based PRNGs
3. Conclusions and open questions

Our goals

Goal: Find a sponge-based PRNG with:

- Forward secrecy + backward secrecy.
- Pseudorandomness for all high-entropy sources
  - *including those that may depend on the permutation.*

SPRG: Our Proposal for Sponge-based PRNGs



- **setup**: sample *seed*
- **refresh:** input whitening using *seed*
- **next**: upper-state zeroing, additional π-call

How to model security?

Robustness notion [DPRVW13] adapted to the random permutation model.

Main ideas:

- The source of weak randomness is also adversarial.
- Incorporates both forward and backward security within same security game!

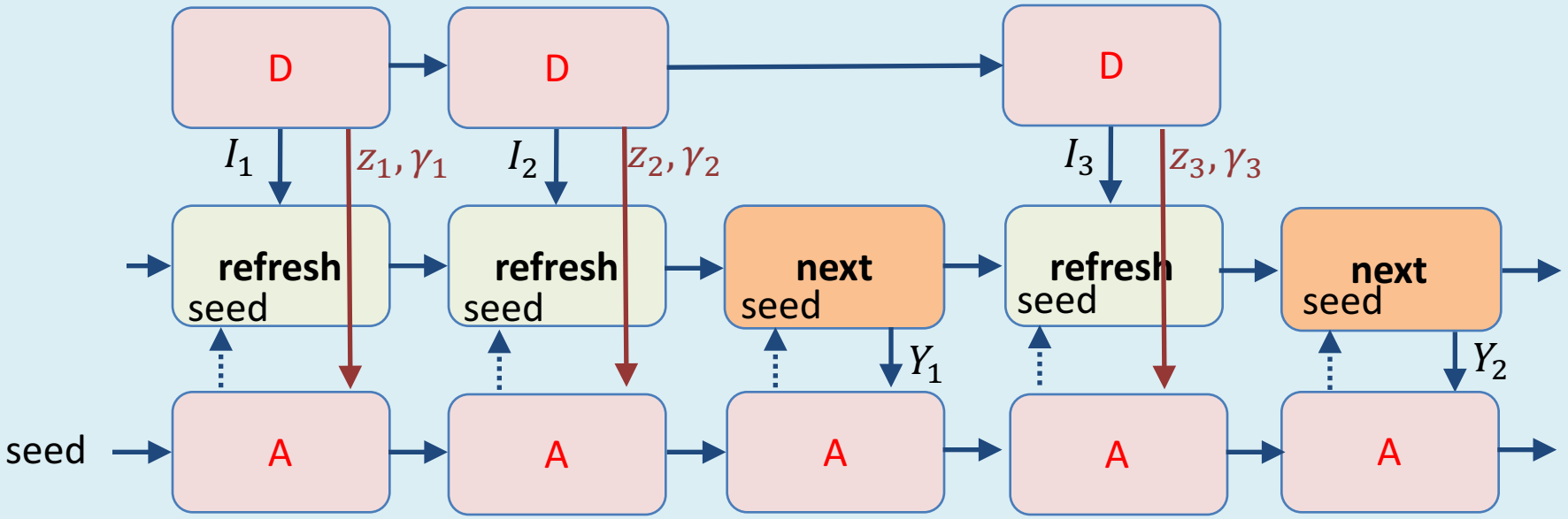
Distribution sampler D

- generates inputs to PRNG
- **legitimate:** provides truthful entropy lower bounds
- does not know seed!

Attacker A

- knows the seed
- can compromise state
- can trigger refresh
- can ask for a **real-or-random** challenge

Robust PRNGs [DPRVW13]



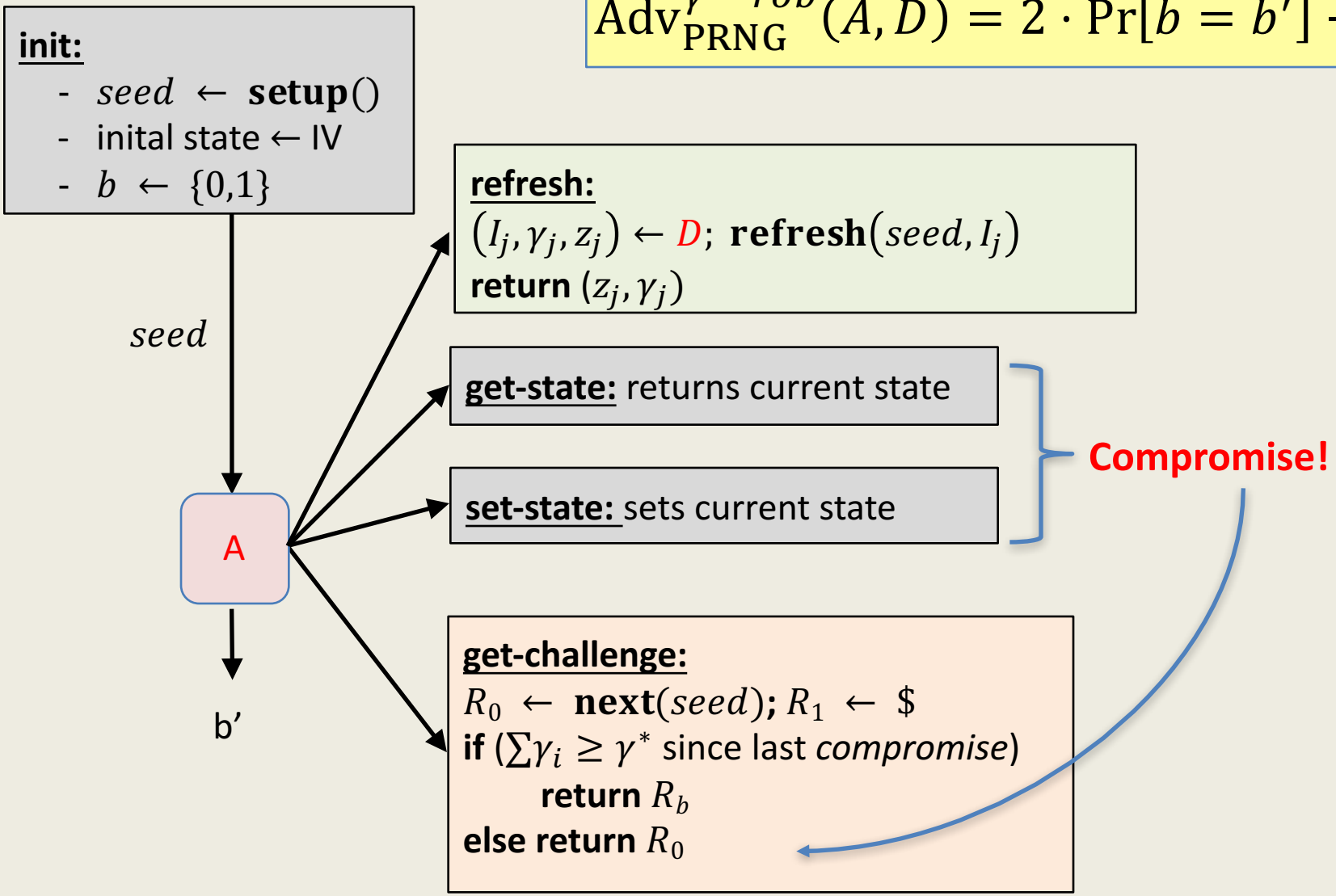
Legitimate sampler:

$$\mathbf{H}_\infty\left(I_j \middle| I_{i\neq j}, z_1, z_2, .., z_k\right) \geq \gamma_j$$

Here: $\mathbf{H}_\infty(X|Y) = \min_y \mathbf{H}_\infty(X|Y = y)$

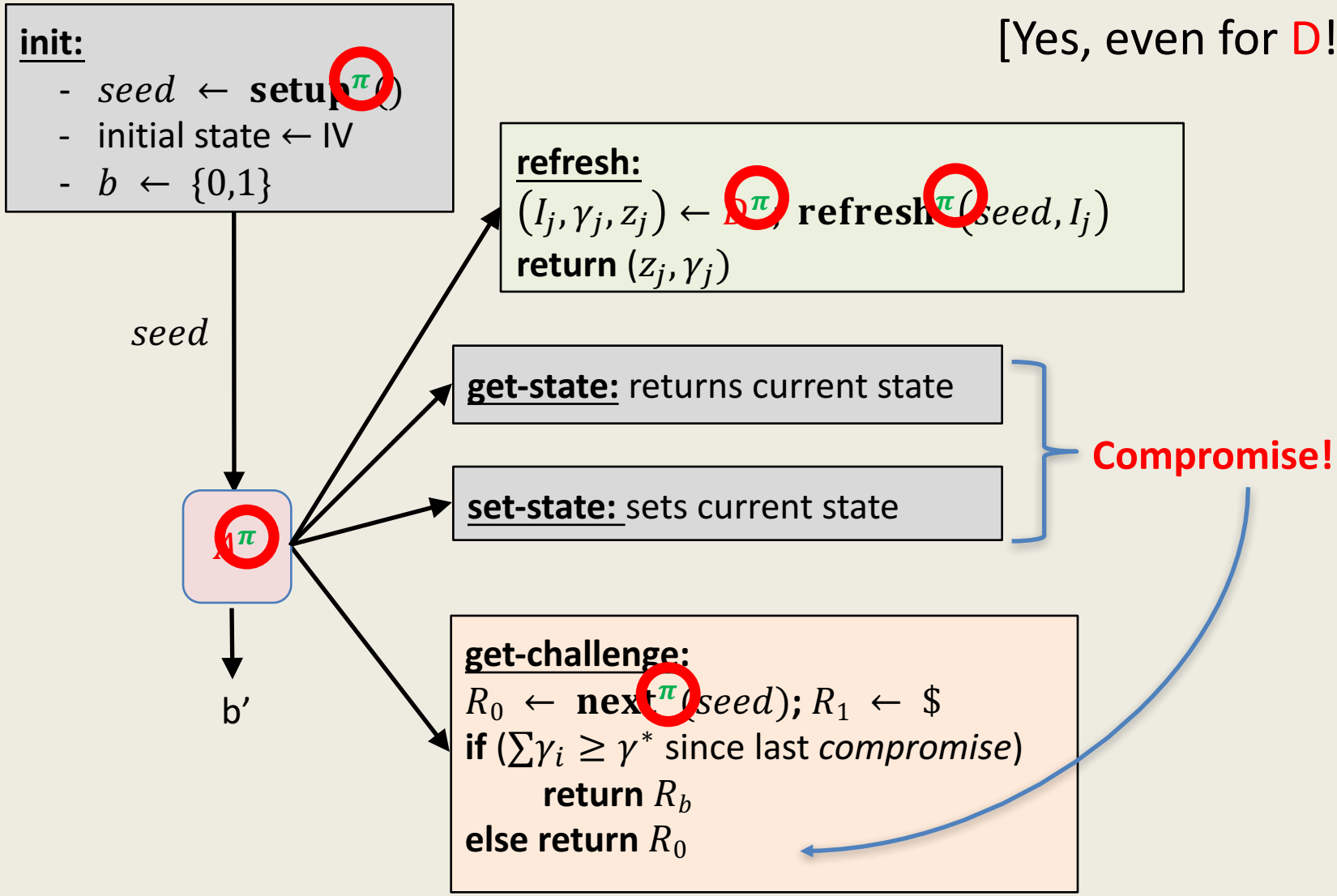
Robust PRNGs [DPRVW13]

$$\mathrm{Adv}^{\gamma^*-rob}_{\mathrm{PRNG}}(A, D) = 2 \cdot \Pr[b = b'] - 1$$



Extension to the Random Permutation Model

Basic idea: Add permutation access for everyone!



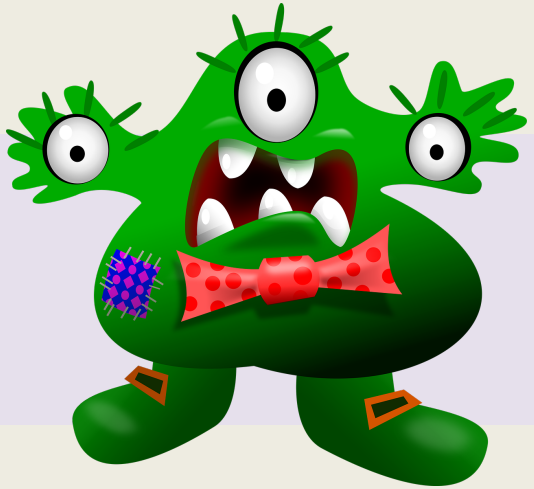
[Yes, even for D!]

RPM Legitimate Samplers

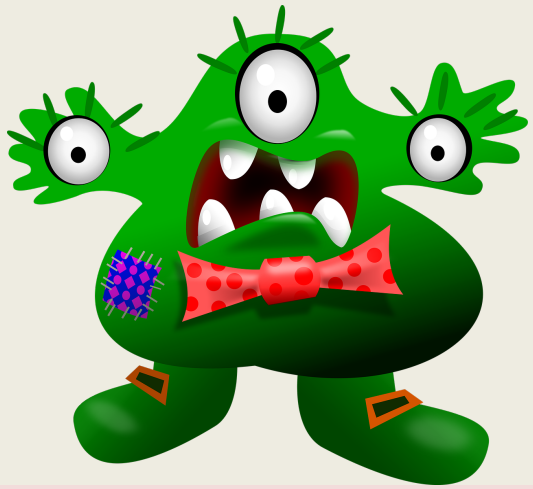
Catch: What does $\mathbf{H}_\infty\left(I_j \middle| I_{i\neq j}, z_1, z_2, .., z_k\right) \geq \gamma_j$ mean in the RPM?

- I_j may be unpredictable only for attackers with bounded queries to $\boldsymbol{\pi}$
- **Example:** $I_j = \boldsymbol{\pi}^k(0^n)$

Current definition of legitimate sampler: A somewhat-unsatisfactory monster!



Legitimate samplers



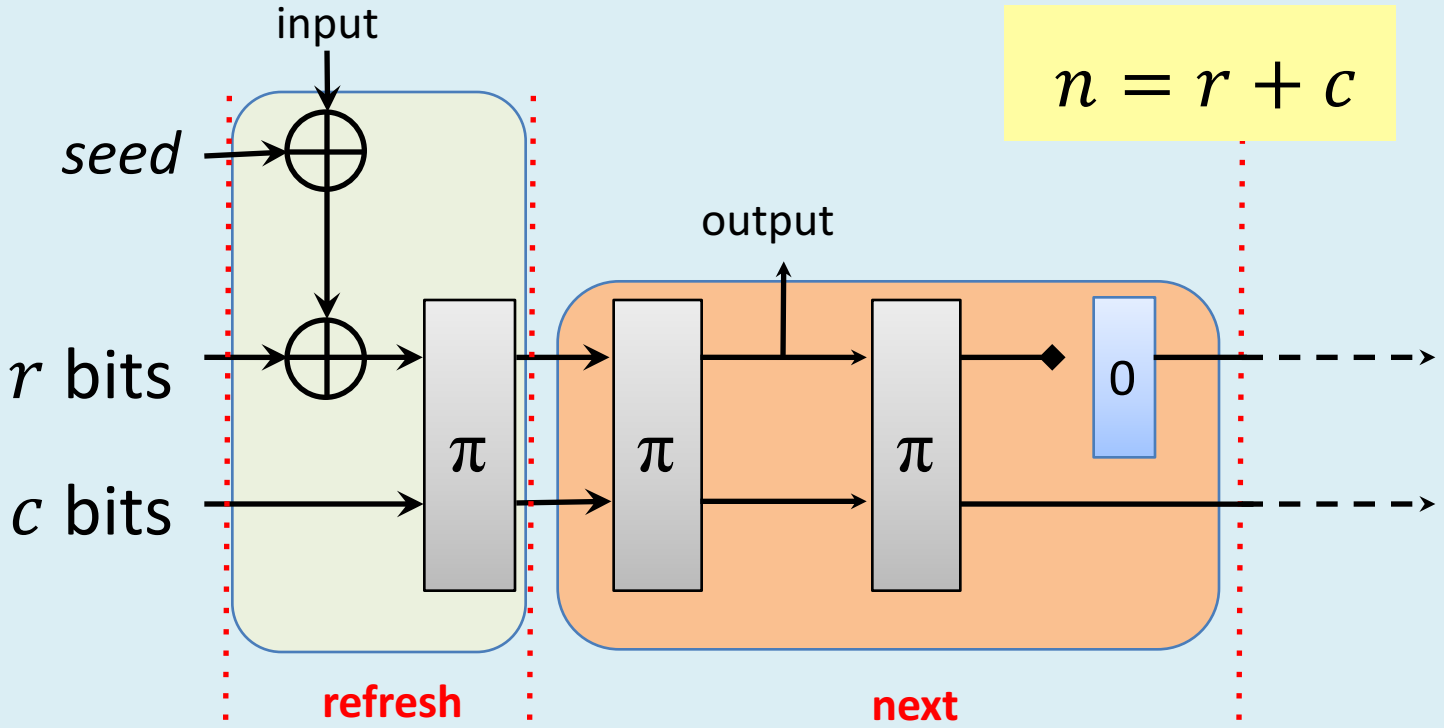
$$\mathbf{H}_\infty\left(I_j \middle| I_{i\neq j}, z_1, z_2, .., z_k\right) \geq \gamma_j$$

"No adversary making q_π queries to π should be able to guess I_j with prob. better than $2^{-\gamma_j}$, even given all I_i for $j \neq i$, $z_1, \ldots, z_k$, and all permutations queries made by D, except those needed to compute I_j"

"q_π-legitimate sampler"

Main Theorem – Robustness

e.g., $n = 1600$, $c \geq 1024$



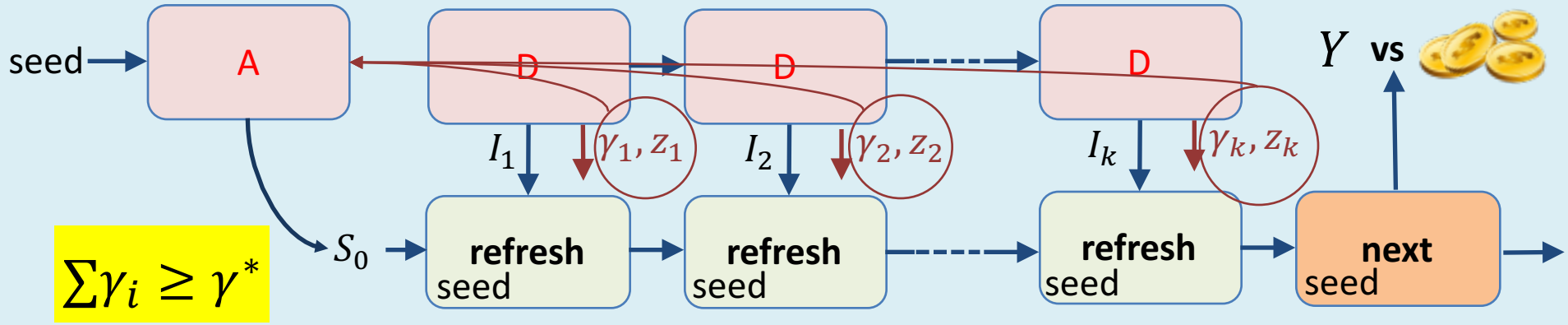
Theorem. [Informal] $\forall D, A$ making $\leq q_\pi$ queries, and A making $\leq q_R$ real-or-random queries:

$$\mathrm{Adv}^{\gamma^*-rob}_{\mathrm{PRNG}}(A, D) \leq q_R \times (\text{something small})$$

As long as: $q_\pi \leq \min\{2^{\gamma^*}, 2^{\frac{c}{2}}, 2^r\}$

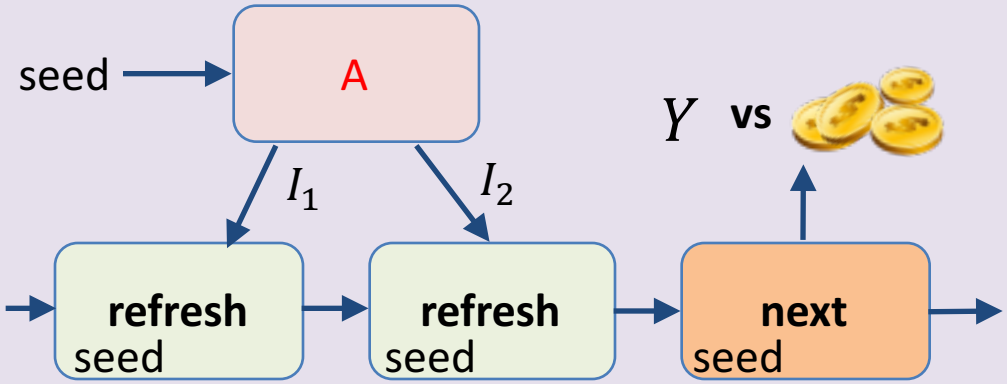
Proof overview – Two Steps

γ^*-recovering security



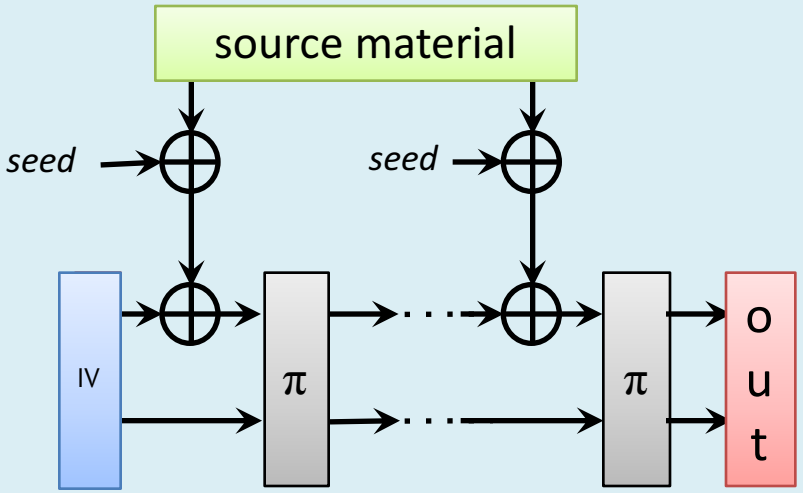
preserving security

If initialized with "good state", output of **next** *is pseudorandom for adversarially chosen* $I_1, I_2, \ldots$

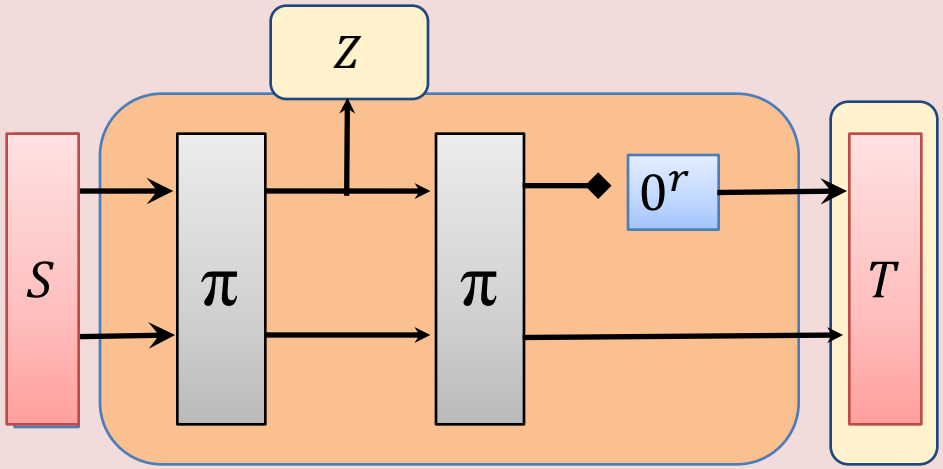


Two key lemmas

“Sponge extraction lemma”



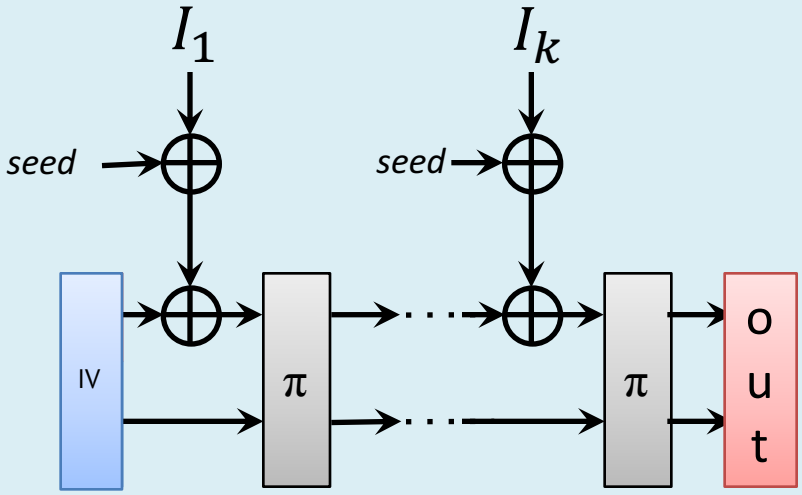
Analysis of next



next

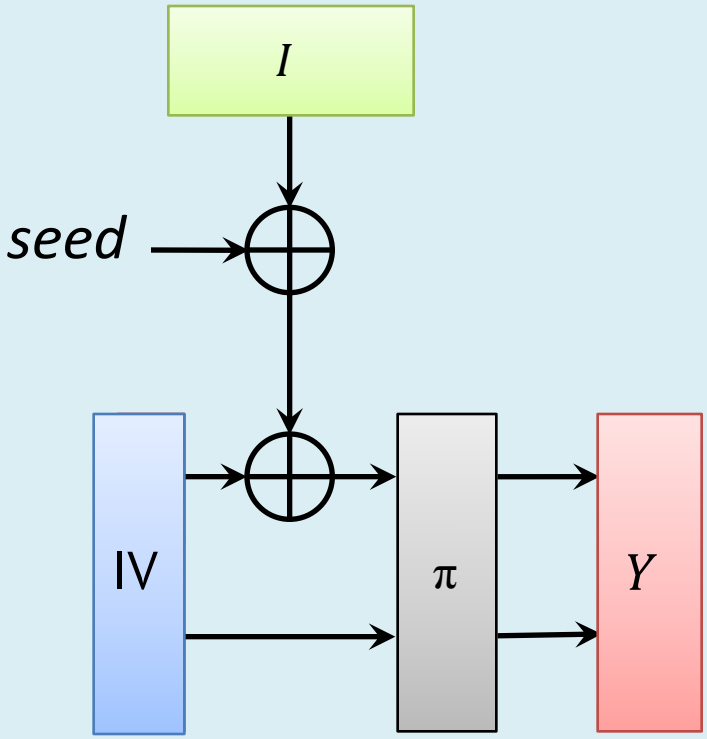
Key Lemma– Sponge Extraction

Key question: Can sponges act as good randomness extractors?



E.g. $(seed, out) \approx (seed, \$)$ if $\mathbf{H}_\infty(I_1 \ldots I_k) \geq \gamma^*$

It depends: One-round case



e.g., imagine source samples

$$I = 0||W$$

where W is a uniform $(r-1)$-bit string.

Distinguisher $D(seed, Y)$:

$T \leftarrow \pi^{-1}(Y)$

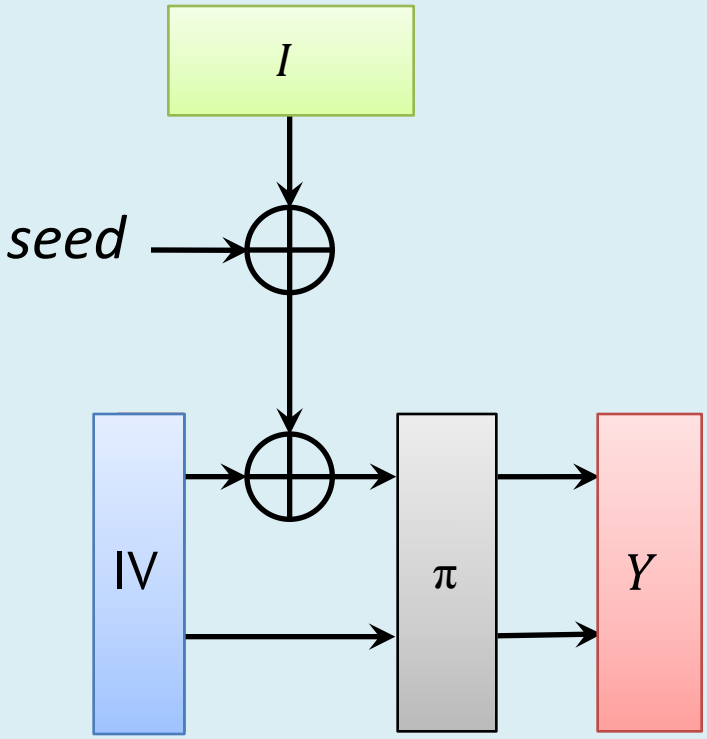
if $T[1] \oplus IV[1] \oplus seed[1] = 0$ **then**

return 1

else return 0

The attack was possible <u>because</u> we have been able to query $\pi^{-1}(Y)$ … so what if we can't?

It depends: One-round case

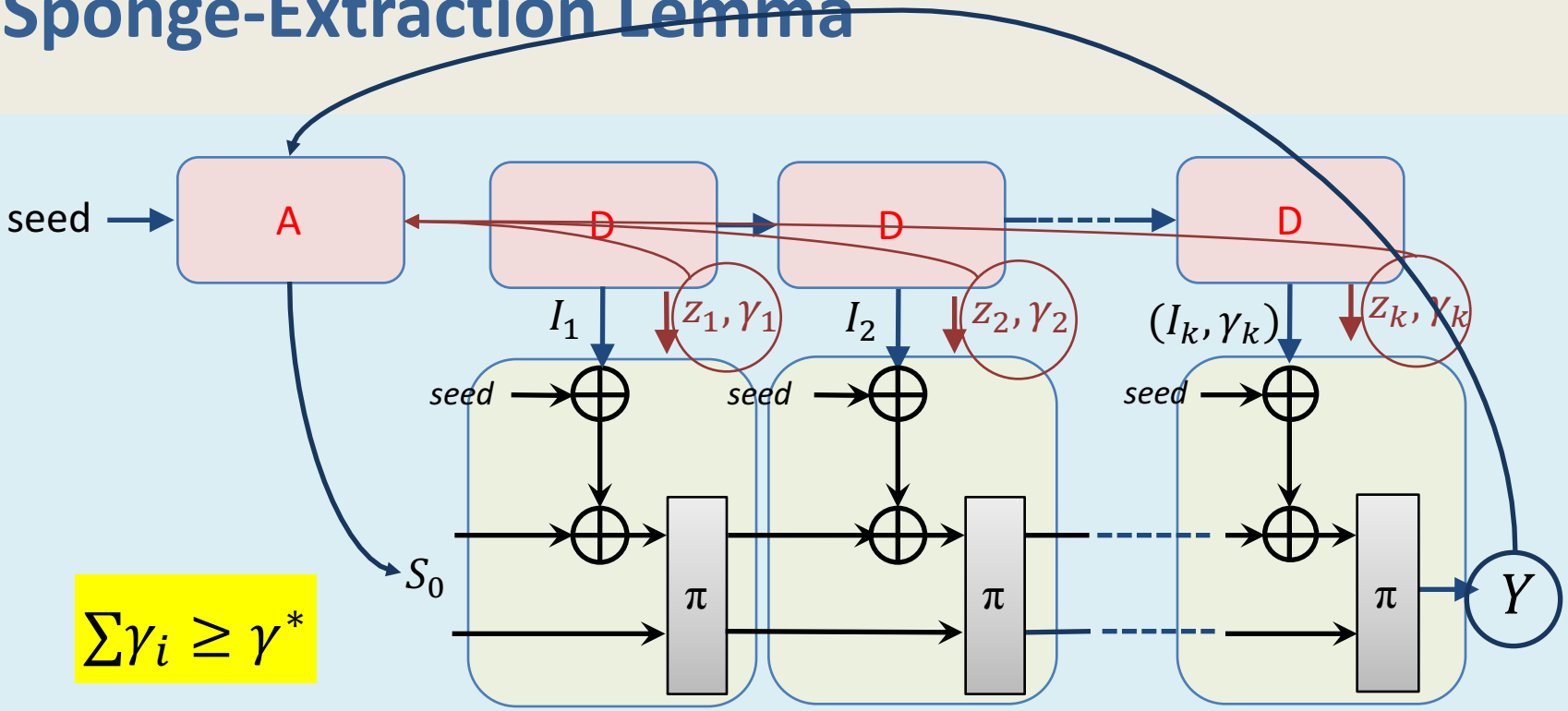


Intuition: If $D(seed, Y)$ cannot query $\pi^{-1}(Y)$, then needs to query $\pi(IV \oplus seed \oplus I)$ on all possible I's!

Work needed to distinguish: $2^{\mathbf{H}_\infty(I)} = 2^{r-1}$ queries to π!

Main observation: Restriction that $\pi^{-1}(Y)$ is never queried is valid in applications where Y is used as a secret key!

Sponge-Extraction Lemma



Lemma 1. Output Y is pseudorandom, provided:

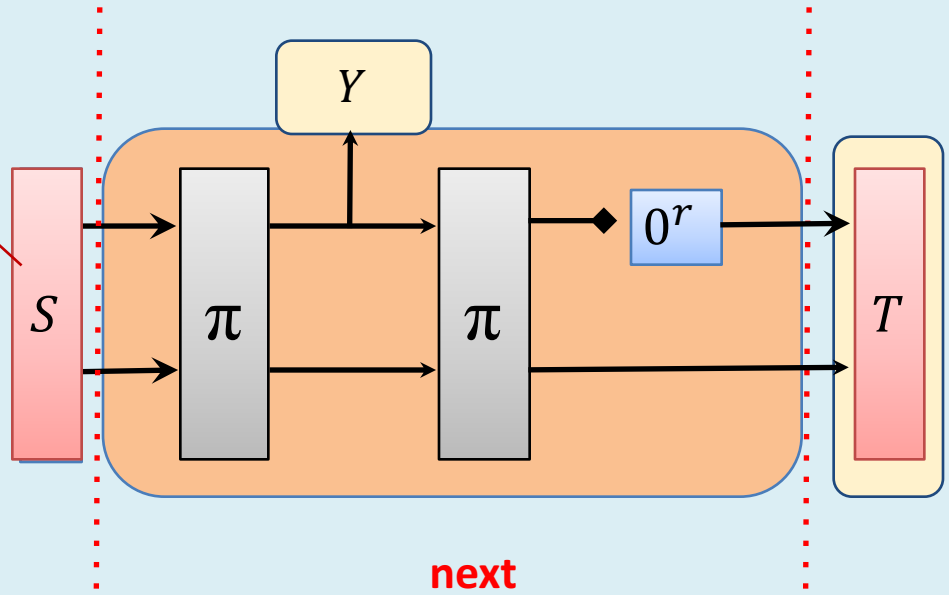
1. $q_\pi \leq \min\{2^{\gamma^*}, 2^{\frac{c}{2}}, 2^r\}$

   here, q_π is # queries by A and D combined!

2. A never queries $\pi^{-1}(Y)$

Key Lemma– Analysis of next

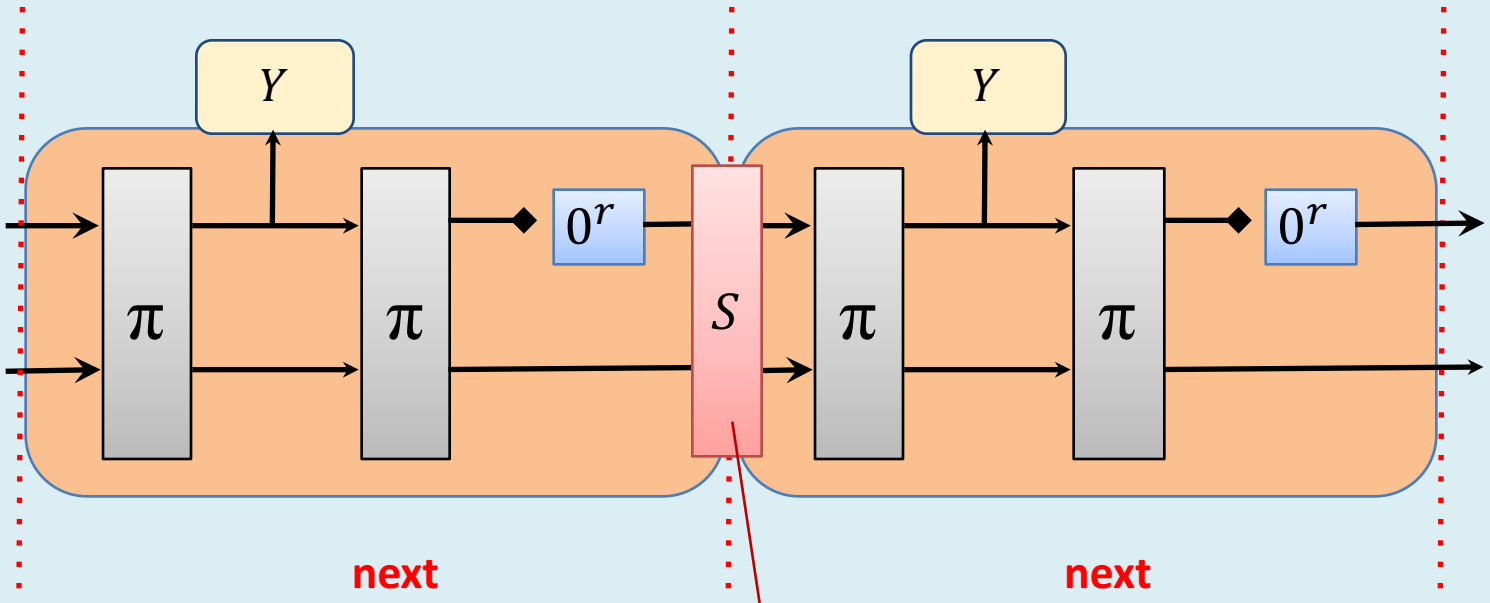
Only assume lower bound on $\mathbf{H}_\infty(S)$



Lemma 2. $(Y, T) \approx (U_r, 0^r || U_c)$ for any distinguisher that makes $q_\pi \ll \min\{2^{\mathbf{H}_\infty(S)}, 2^r, 2^{c/2}\}$ queries to π.

Next – Remarks

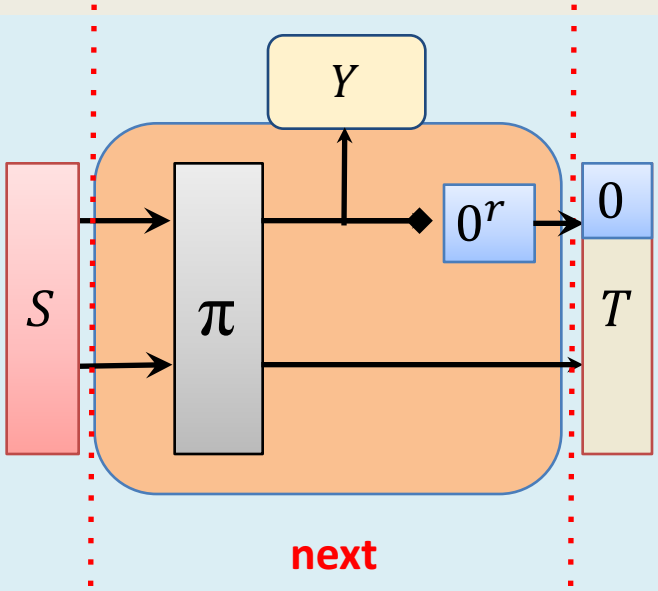
General distribution on S is necessary, as we may call **next** multiple times!



$$\approx 0^r || U_c$$

Next – Remarks (cont'd)

Extra permutation call is necessary!

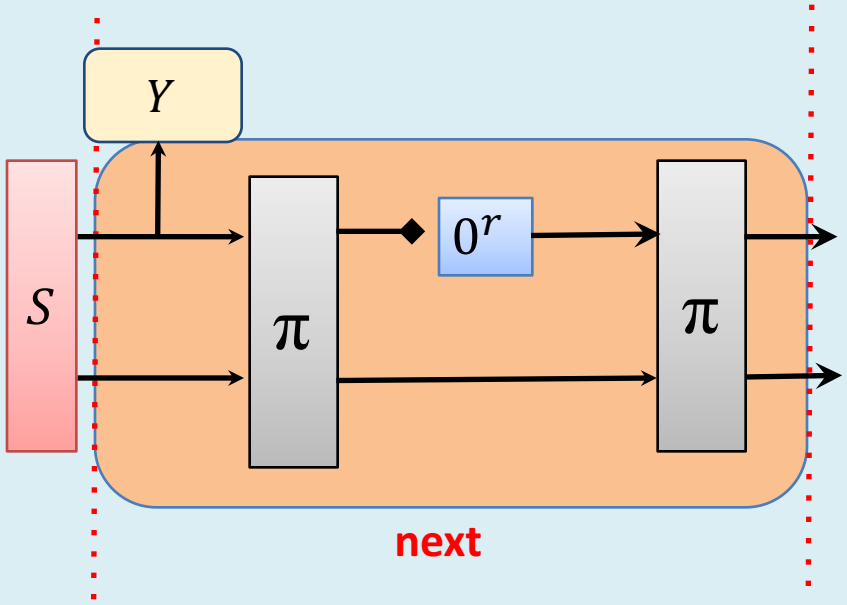


Attacker just checks whether $\pi^{-1}(Y||T)$ is in the range of S

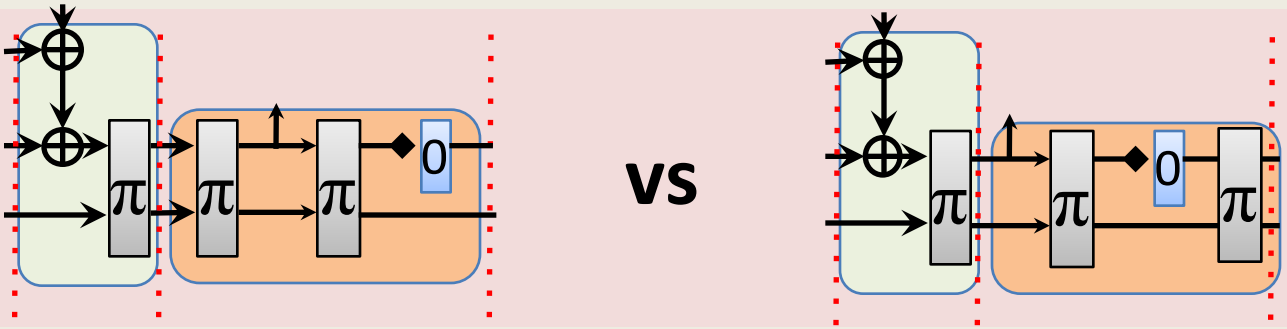
Note: Extra cost of additional permutation call can be mitigated by outputting multiple Y's.

Alternative – Open question

Following variant does not fit into our proof framework, but may be fine overall.

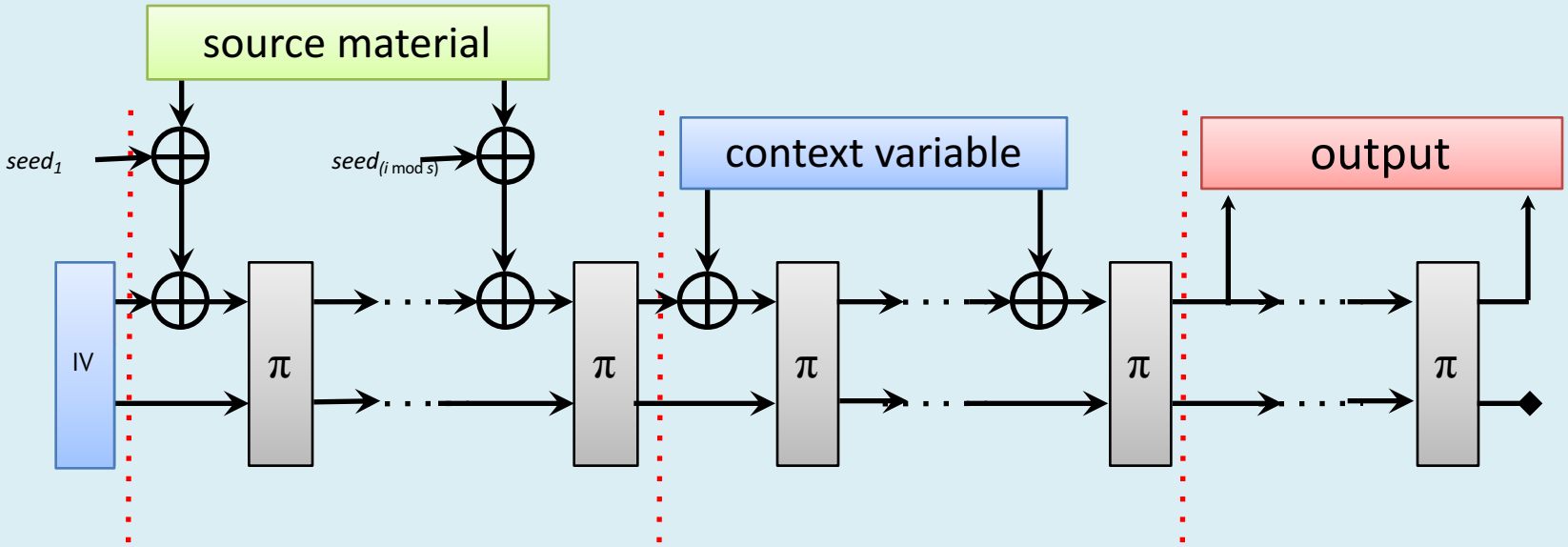


This would prevent a double permutation call when transitioning refresh -> next!



[Hutchinson, SAC '16] proposes another approach to next, requires modification of lower bits!

Further application – Sponge-based KDF



We show it is a good KDF, even when source material is permutation dependent!

Proof combines sponge extraction lemma + PRF analyses for keyed sponges [ADMvA15,GPT15,MRV15]

Roadmap of this talk

1. PRNGs: Sponge-based Instantiations
2. Provably-robust sponge-based PRNGs
3. Conclusions and open questions

Permutation-dependence and the seed: Why care?

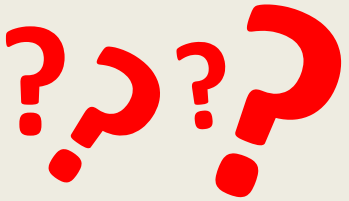
Typical argument: Real-world distributions behave nicely!

Possible, but … it is not easy to characterize what “real-world distribution” means...

Personal take: *If you can add security for cheap, then why not enable it as an option?*

Our seeding is entirely black box – input whitening!

Open Problems



- **Better security**
  - Premature next? More general class of samplers?
- **Concrete bounds**
  - No issue for large-stage ($n = 1600$ bits)
- **Small state**
  - What if state is very small (e.g., 128 bits) and randomness is injected at low rate
  - Incorrect proposal in our paper ☹
- **Assumptions**
  - Random permutation should make things easier, except it does not!

Open Problems – Assumptions

Random-permutation assumption problematic

- Possible way out: Public-seed PRPs [Soni-**T**, EC'17]
  - Standard-model assumption for (seeded) permutations
- Caveat: Permutation itself requires a seed!
  - See Pratik's talk on Wednesday [not about PRNGs]

Thank you!