A Robust and Sponge–Like PRNG With Improved Efficiency

Daniel Hutchinson

Information Security Group,
Royal Holloway, University of London

August 11, 2016

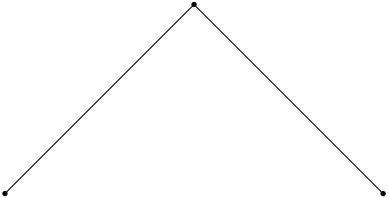
ToC

1. Preliminaries
2. Security notions
3. Designs
4. Conclusions

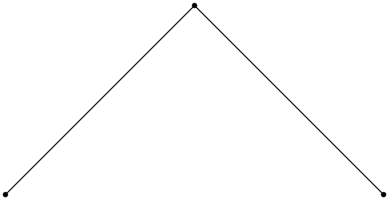
Randomness

•

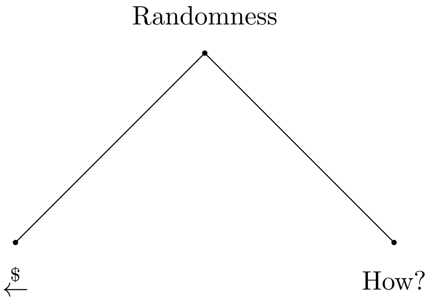
Randomness



Randomness



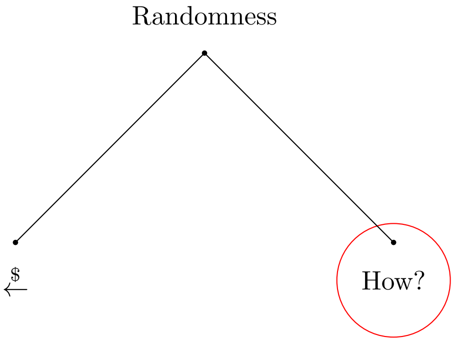
$\overset{\$}{\leftarrow}$



Randomness

$\xleftarrow{\$}$

How?



Randomness

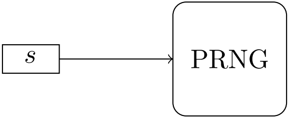
$\xleftarrow{\$}$

How?

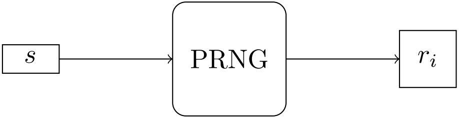
PRNG with Input

PRNG

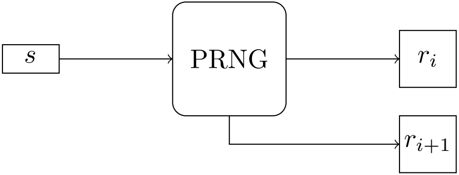
PRNG with Input



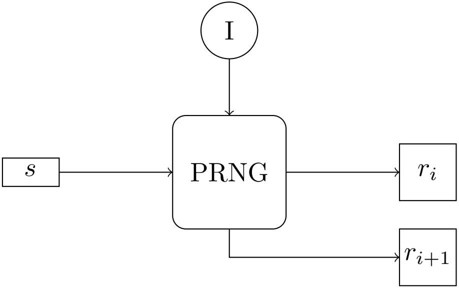
PRNG with Input



PRNG with Input



PRNG with Input

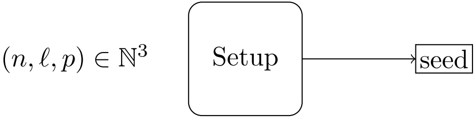


PRNG with Input

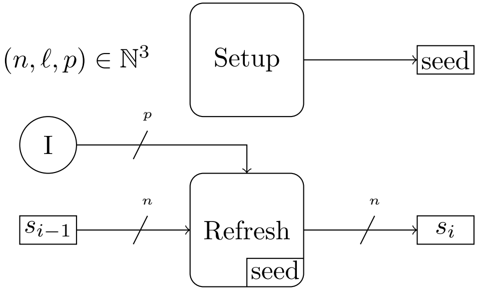
WARNING

Seed in this definition is public, NOT the initial state.

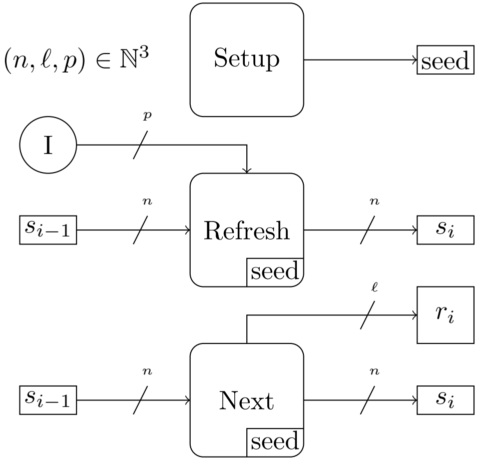
PRNG with Input



PRNG with Input



PRNG with Input



Security notions for PRNGs with input[1]

- Resilience - RES ←Weakest notion
  Basic security, no compromise of state, looks random.
- Forward security - FWD
  Output still random even if state is compromised afterwards.
- Backward security - BWD
  Output looks random even if state is compromised previously, but enough entropy has been input since then.
- Robustness - ROB ←Strongest notion
  Combination of the above; adversary can tamper with state.

[1] Yevgeniy Dodis et al. (2013). *Security Analysis of Pseudo-Random Number Generators with Input: /dev/random is not Robust*.

Security notions for PRNGs with input[1]

- Resilience - RES ←Weakest notion
  Basic security, no compromise of state, looks random.

- Forward security - FWD
  Output still random even if state is compromised afterwards.

- Backward security - BWD
  Output looks random even if state is compromised previously, but enough entropy has been input since then.

- Robustness - ROB ←Strongest notion
  Combination of the above; adversary can tamper with state.

[1] Yevgeniy Dodis et al. (2013). *Security Analysis of Pseudo-Random Number Generators with Input: /dev/random is not Robust*.

Security notions for PRNGs with input[1]

- Resilience - RES ←Weakest notion
  Basic security, no compromise of state, looks random.
- Forward security - FWD
  Output still random even if state is compromised afterwards.
- Backward security - BWD
  Output looks random even if state is compromised previously, but enough entropy has been input since then.
- Robustness - ROB ←Strongest notion
  Combination of the above; adversary can tamper with state.

[1] Yevgeniy Dodis et al. (2013). *Security Analysis of Pseudo-Random Number Generators with Input: /dev/random is not Robust*.

Security notions for PRNGs with input[1]

- Resilience - RES ←Weakest notion
  Basic security, no compromise of state, looks random.
- Forward security - FWD
  Output still random even if state is compromised afterwards.
- Backward security - BWD
  Output looks random even if state is compromised previously, but enough entropy has been input since then.
- Robustness - ROB ←Strongest notion
  Combination of the above; adversary can tamper with state.

[1] Yevgeniy Dodis et al. (2013). *Security Analysis of Pseudo-Random Number Generators with Input: /dev/random is not Robust*.

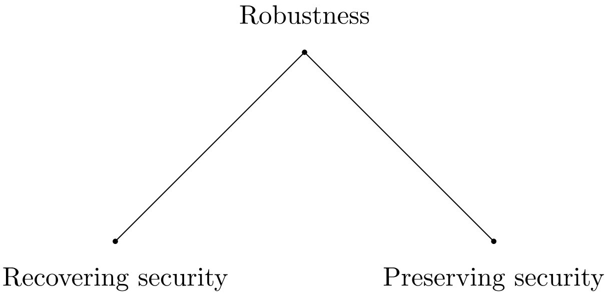
Security notions for PRNGs with input[1]

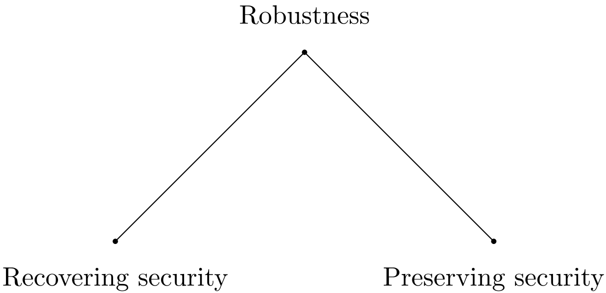
- Resilience - RES ←Weakest notion
  Basic security, no compromise of state, looks random.
- Forward security - FWD
  Output still random even if state is compromised afterwards.
- Backward security - BWD
  Output looks random even if state is compromised previously, but enough entropy has been input since then.
- Robustness - ROB ←Strongest notion
  Combination of the above; adversary can tamper with state.

[1] Yevgeniy Dodis et al. (2013). *Security Analysis of Pseudo-Random Number Generators with Input: /dev/random is not Robust*.

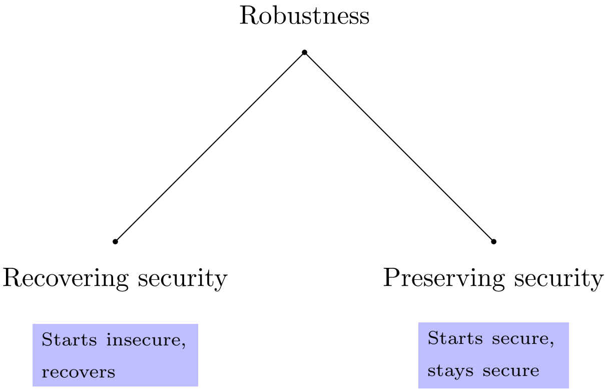
Robustness

-


Robustness
Recovering security
Preserving security



Starts insecure, recovers



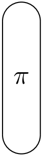
Robustness

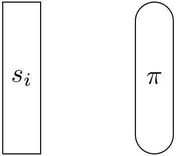
Recovering security

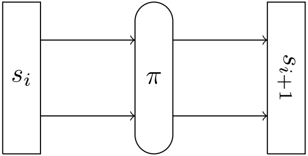
Starts insecure, recovers

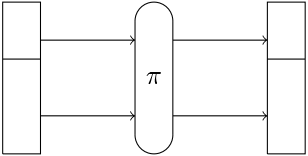
Preserving security

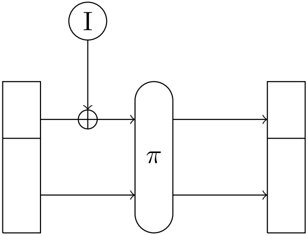
Starts secure, stays secure

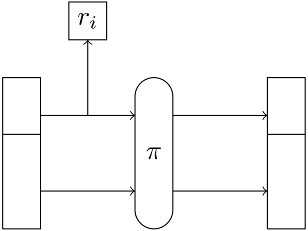

π

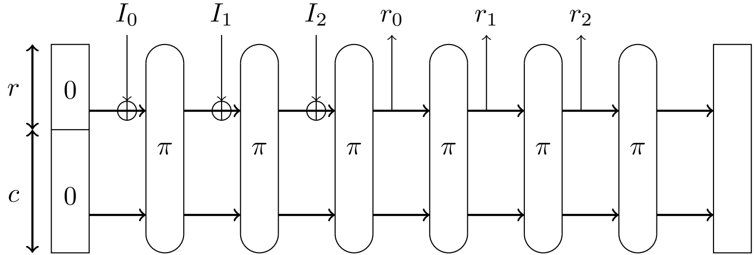

s_i
π


s_i
π
s_{i+1}

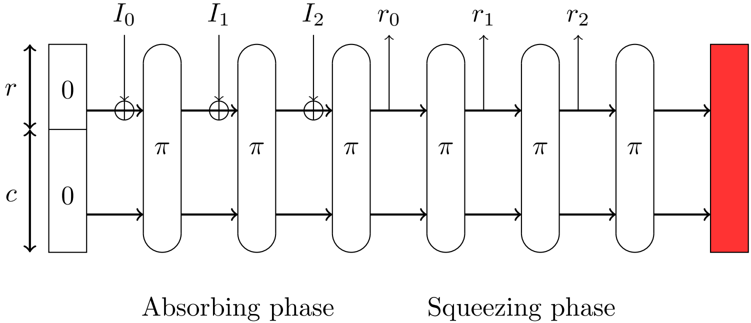

π

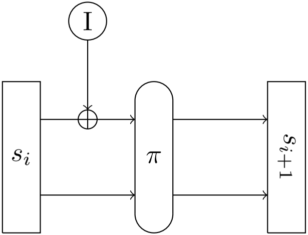

I
π


r_i
π

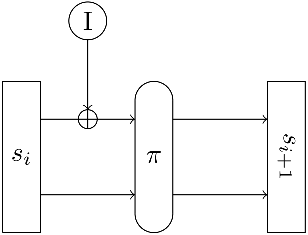


Absorbing phase Squeezing phase

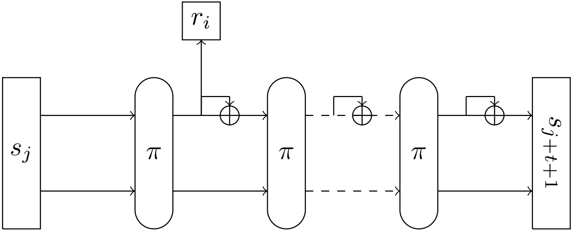

I_0
I_1
I_2
r_0
r_1
r_2
r
c
0
0
π
π
π
π
π
π
Absorbing phase
Squeezing phase



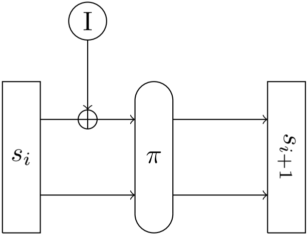
spongeprng.refresh



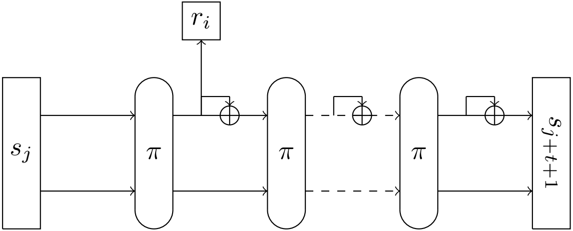
spongeprng.refresh



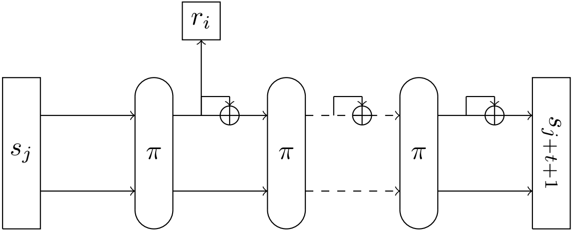
spongeprng.next



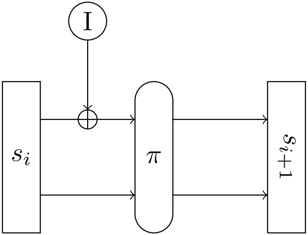
spongeprng.refresh



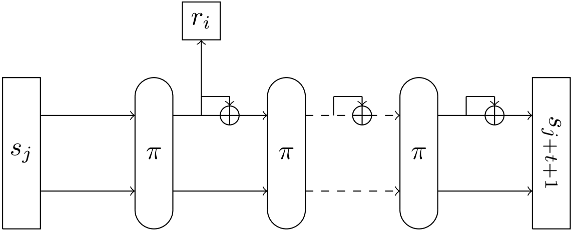
spongeprng.next



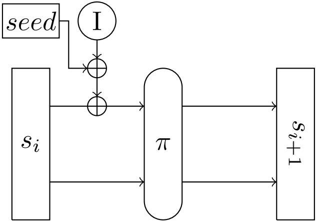
SPRG.next $\boxed{\text{IPM}}$



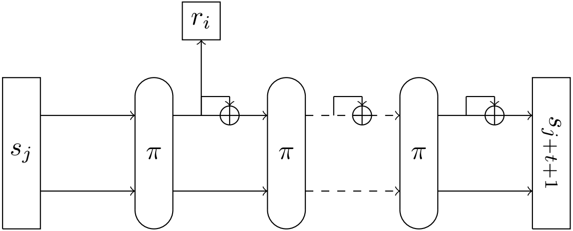
spongeprng.refresh



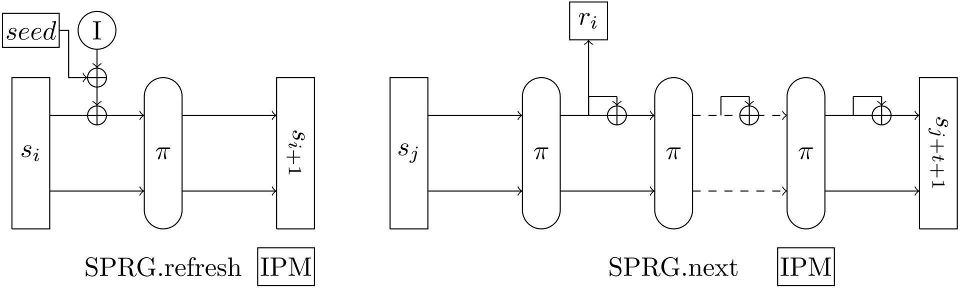
spongeprng.next

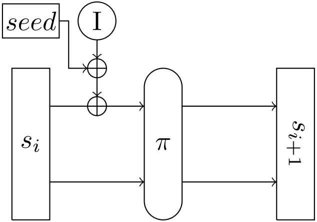


SPRG.refresh $\boxed{\text{IPM}}$

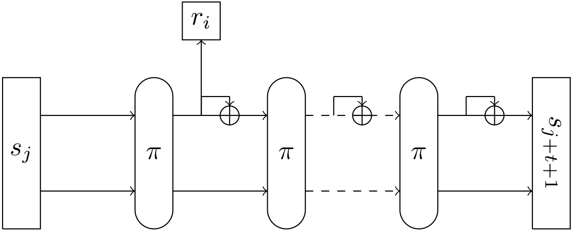


SPRG.next $\boxed{\text{IPM}}$

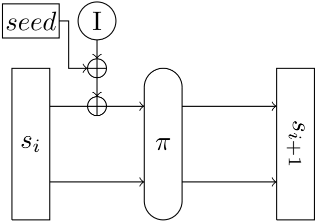

$seed$
I
s_i
π
s_{i+1}
SPRG.refresh
IPM
r_i
s_j
π
π
π
s_{j+t+1}
SPRG.next
IPM



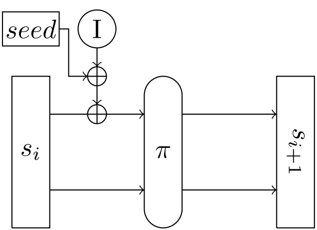
SPRG.refresh $\boxed{\text{IPM}}$



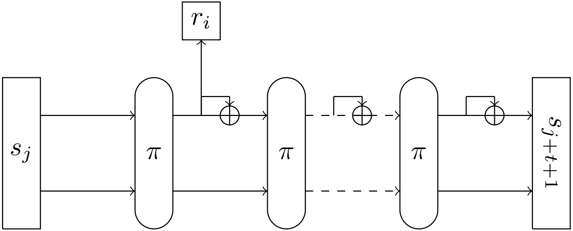
SPRG.next $\boxed{\text{IPM}}$



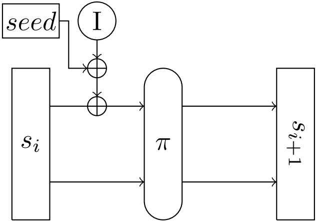
Reverie.refresh $\boxed{\text{IPM}}$



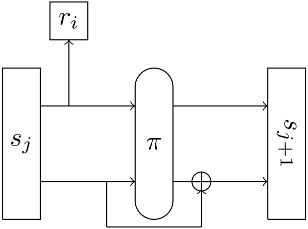
SPRG.refresh IPM



SPRG.next IPM



Reverie.refresh IPM



Reverie.next IPM

Patarin's H-coefficient technique

- Two experiments, real and ideal.

Patarin's H-coefficient technique

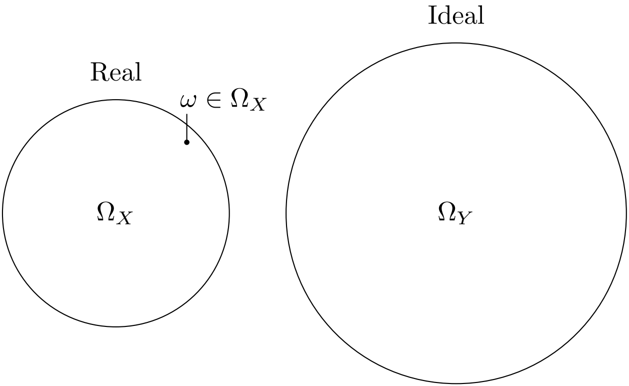
- Two experiments, real and ideal.
- An experiment is described by an oracle ω together with a transcript τ obtained by interacting with ω.

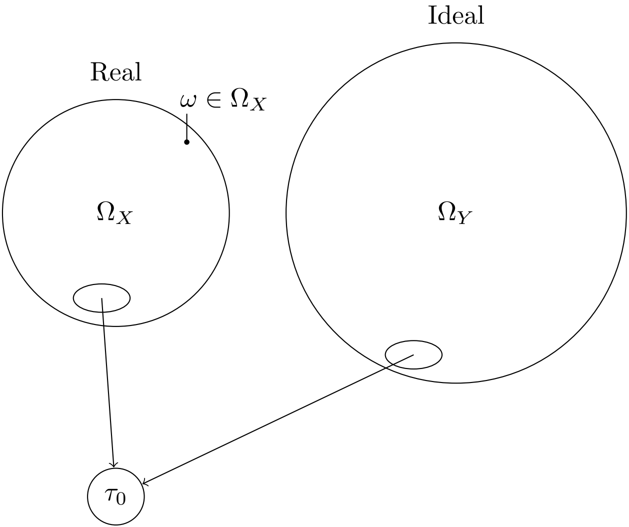
Patarin's H-coefficient technique

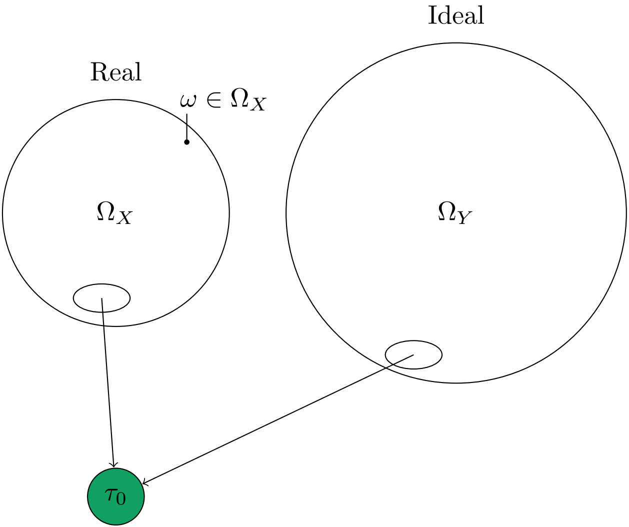
- Two experiments, real and ideal.
- An experiment is described by an oracle ω together with a transcript τ obtained by interacting with ω.
- Ω_X the space of real oracles, while Ω_Y is the space of ideal oracles.

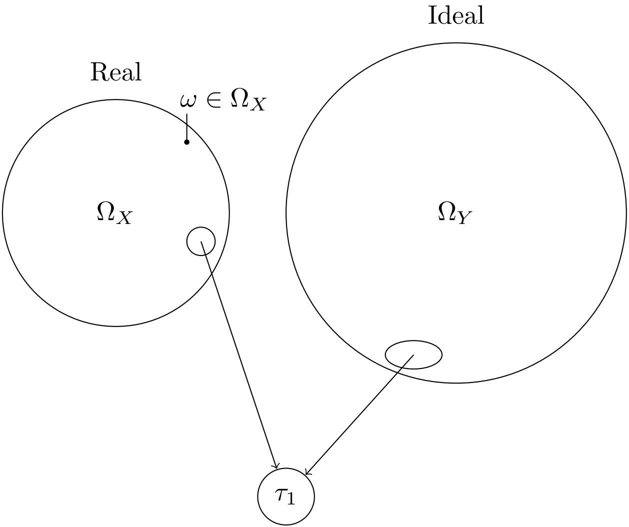
Patarin's H-coefficient technique

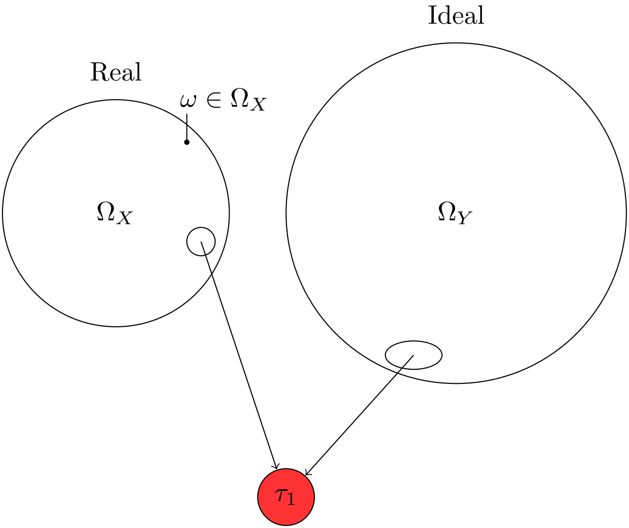
- Two experiments, real and ideal.
- An experiment is described by an oracle ω together with a transcript τ obtained by interacting with ω.
- Ω_X the space of real oracles, while Ω_Y is the space of ideal oracles.
- Transcripts are partitioned into two sets; good or bad.

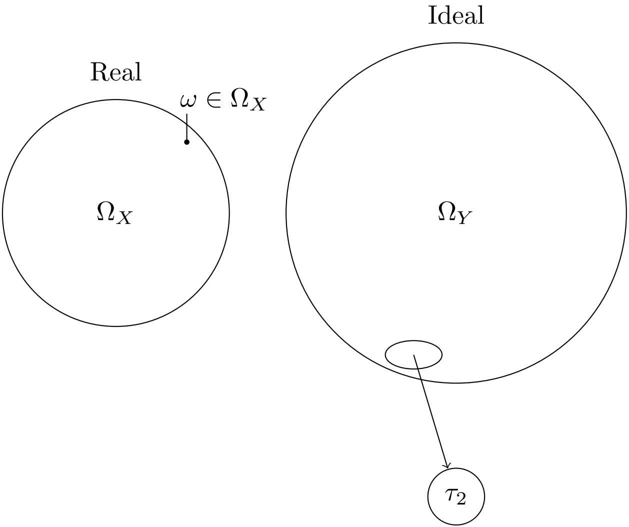

Real
$\omega \in \Omega_X$
Ω_X
Ideal
Ω_Y

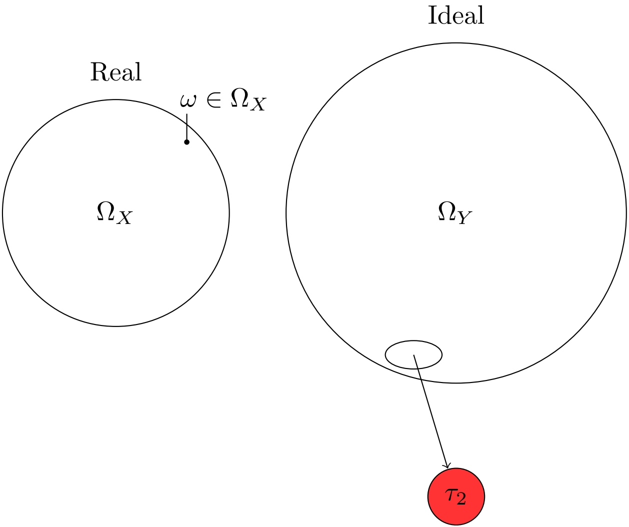

Ideal
Real
$\omega \in \Omega_X$
Ω_X
Ω_Y
τ_0

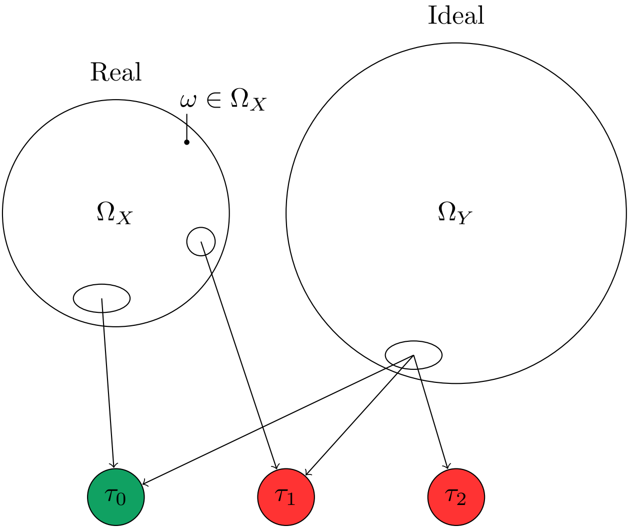

Real
$\omega \in \Omega_X$
Ω_X
Ideal
Ω_Y
τ_0


Ideal
Real
$\omega \in \Omega_X$
Ω_X
Ω_Y
τ_1


Ideal
Real
$\omega \in \Omega_X$
Ω_X
Ω_Y
τ_1


Real
$\omega \in \Omega_X$
Ω_X
Ideal
Ω_Y
τ_2


Real
$\omega \in \Omega_X$
Ω_X
Ideal
Ω_Y
τ_2


Ideal
Real
$\omega \in \Omega_X$
Ω_X
Ω_Y
τ_0
τ_1
τ_2

Conclusions

- Provably robust PRNG design using the H-coefficient technique

Conclusions

- Provably robust PRNG design using the H-coefficient technique

- Design is more efficient than other offerings

Questions

Thank you for listening.

Any Questions?

© Steven Benjamin / Barcroft Media