Backdooring PRGs and PRNGs

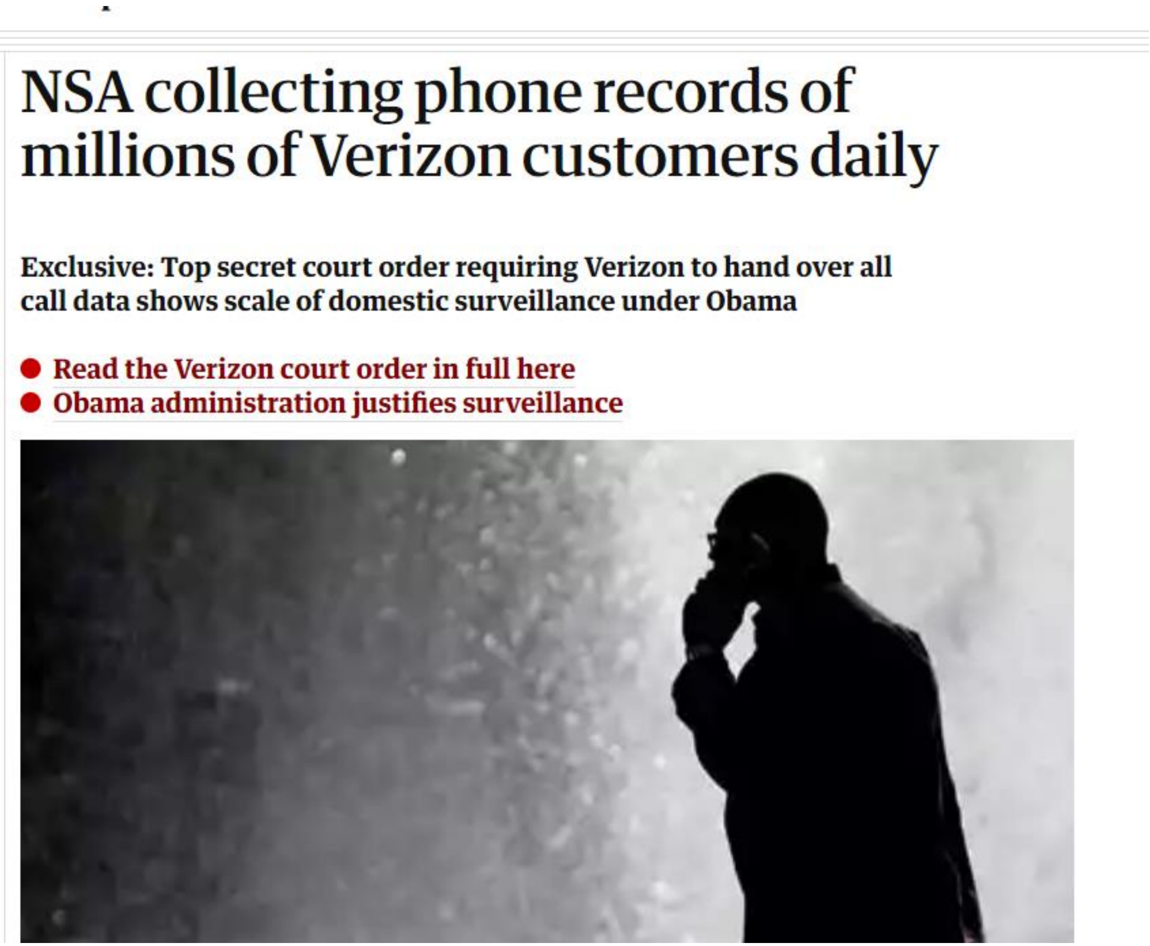
Arka Rai Choudhuri

Snowden Leaks

https://commons.wikimedia.org/wiki/File:Edward_Snowden-2.jpg

Snowden Leaks

https://commons.wikimedia.org/wiki/File:Edward_Snowden-2.jpg



NSA collecting phone records of millions of Verizon customers daily

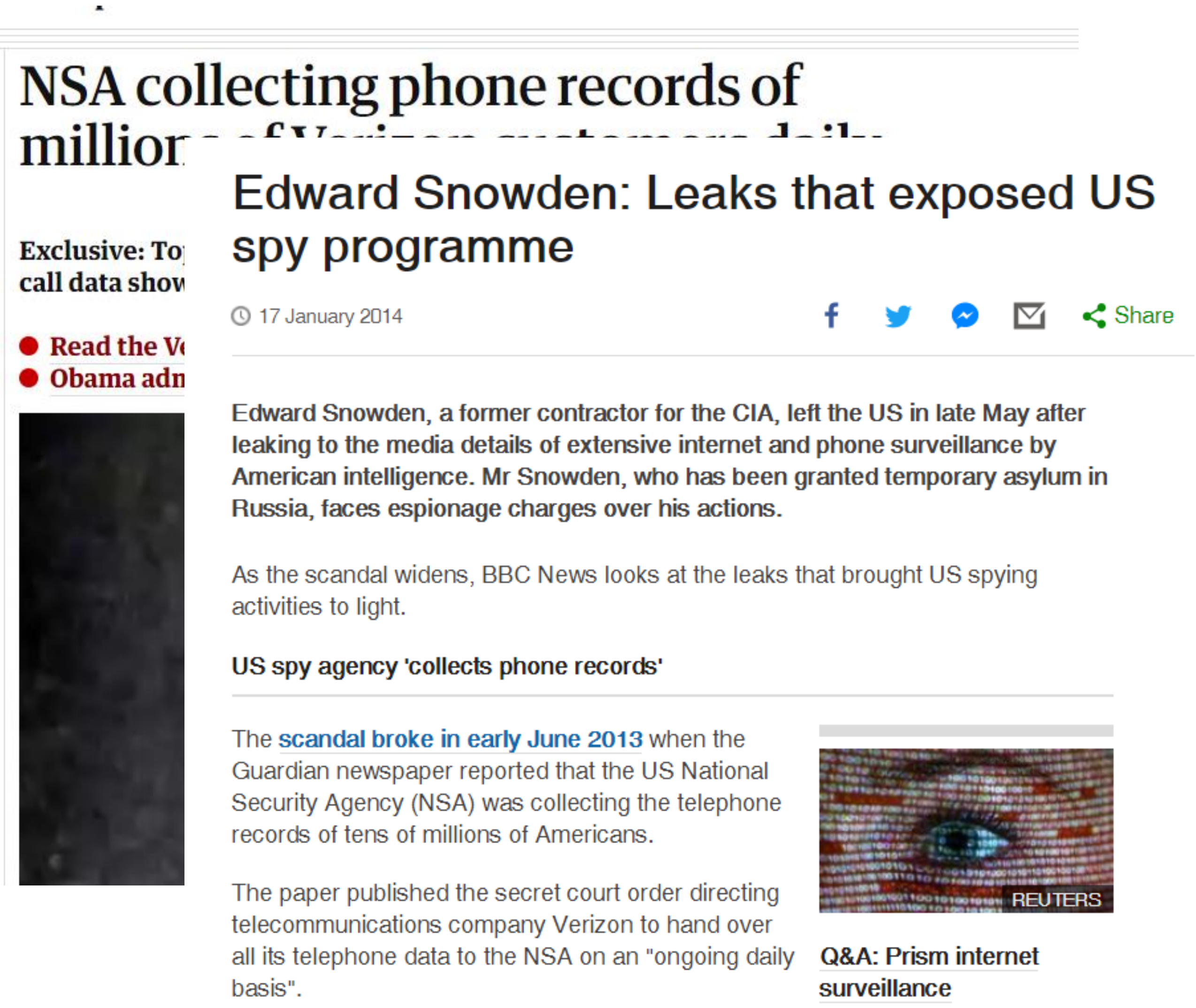
Exclusive: Top secret court order requiring Verizon to hand over all call data shows scale of domestic surveillance under Obama

- Read the Verizon court order in full here
- Obama administration justifies surveillance

The Guardian, June 2013

Snowden Leaks

https://commons.wikimedia.org/wiki/File:Edward_Snowden-2.jpg



NSA collecting phone records of millior

Exclusive: To
call data show

● Read the Ve
● Obama adn

Edward Snowden: Leaks that exposed US spy programme

17 January 2014

Share

Edward Snowden, a former contractor for the CIA, left the US in late May after leaking to the media details of extensive internet and phone surveillance by American intelligence. Mr Snowden, who has been granted temporary asylum in Russia, faces espionage charges over his actions.

As the scandal widens, BBC News looks at the leaks that brought US spying activities to light.

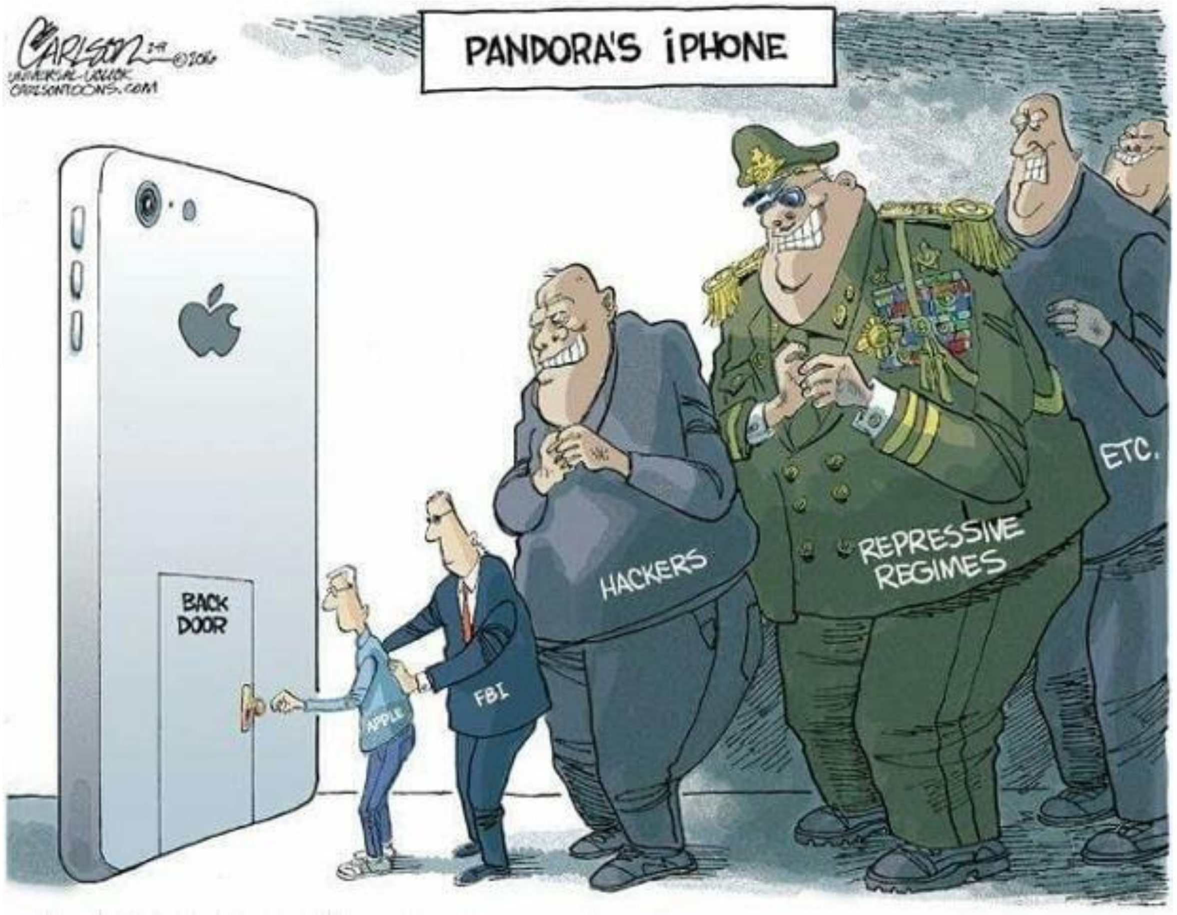
US spy agency 'collects phone records'

The scandal broke in early June 2013 when the Guardian newspaper reported that the US National Security Agency (NSA) was collecting the telephone records of tens of millions of Americans.

The paper published the secret court order directing telecommunications company Verizon to hand over all its telephone data to the NSA on an "ongoing daily basis".

Q&A: Prism internet surveillance

BBC, January 2014



Trap Door by S. Carlson

BIG BROTHER IS
WATCHING YOU

https://commons.wikimedia.org/wiki/File:Cropped-big-brother-is-watching-1984.png

Ubiquity of PRGs and PRNGs with input

Good randomness essentially for a lot of cryptography

- IV, key generation, selection of DH exponents

Ubiquity of PRGs and PRNGs with input

Good randomness essentially for a lot of cryptography

- IV, key generation, selection of DH exponents

Randomness failures have led to vulnerabilities in deployed systems.

Ubiquity of PRGs and PRNGs with input

Good randomness essentially for a lot of cryptography

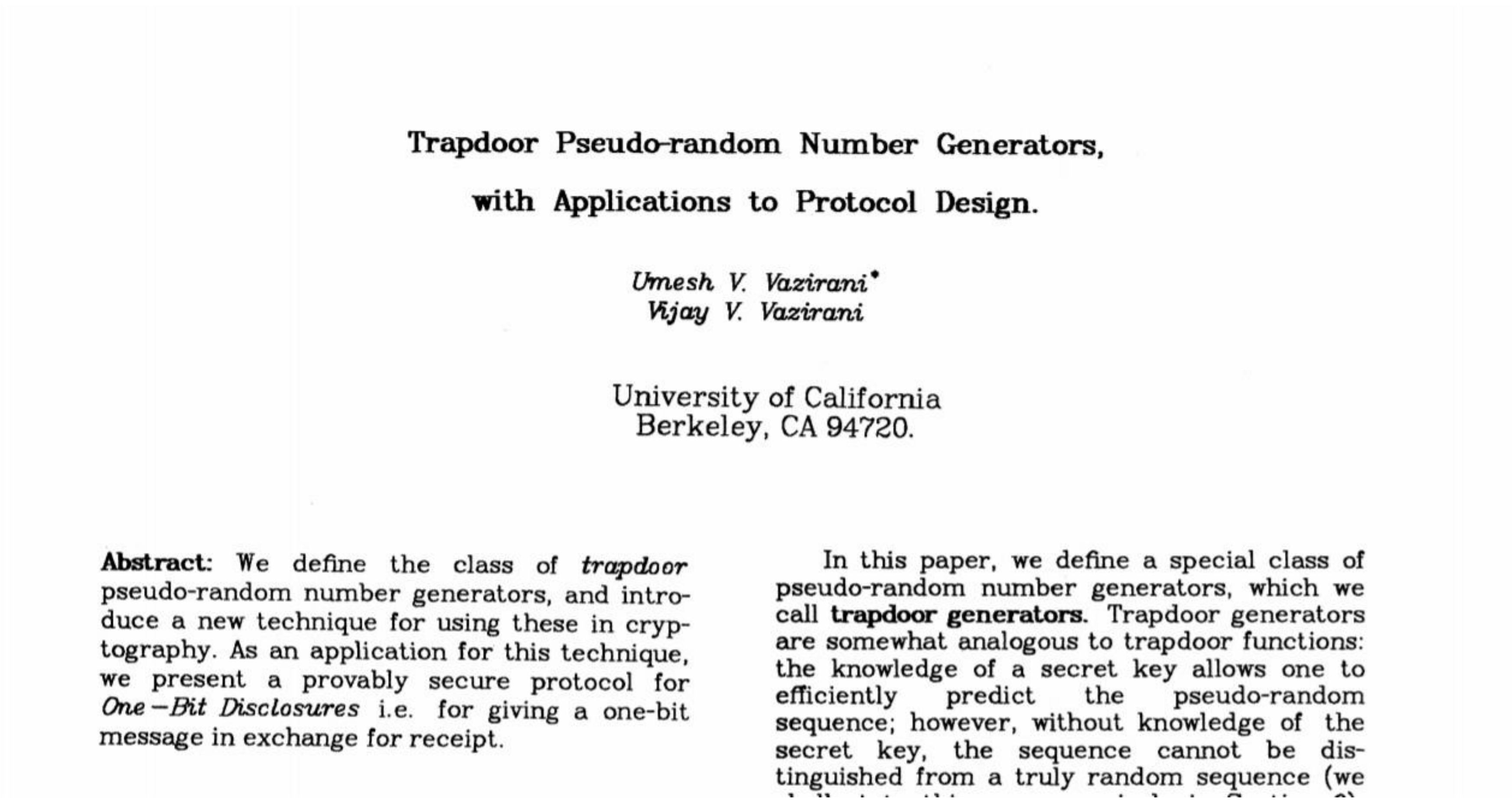
- IV, key generation, selection of DH exponents

Randomness failures have led to vulnerabilities in deployed systems.

Even theory says doing cryptography with bad randomness is not a good idea.

Why do we even care about building these backdoors?

Umesh Vazirani, Vijay Vazirani FOCS 1983



Trapdoor Pseudo-random Number Generators, with Applications to Protocol Design.

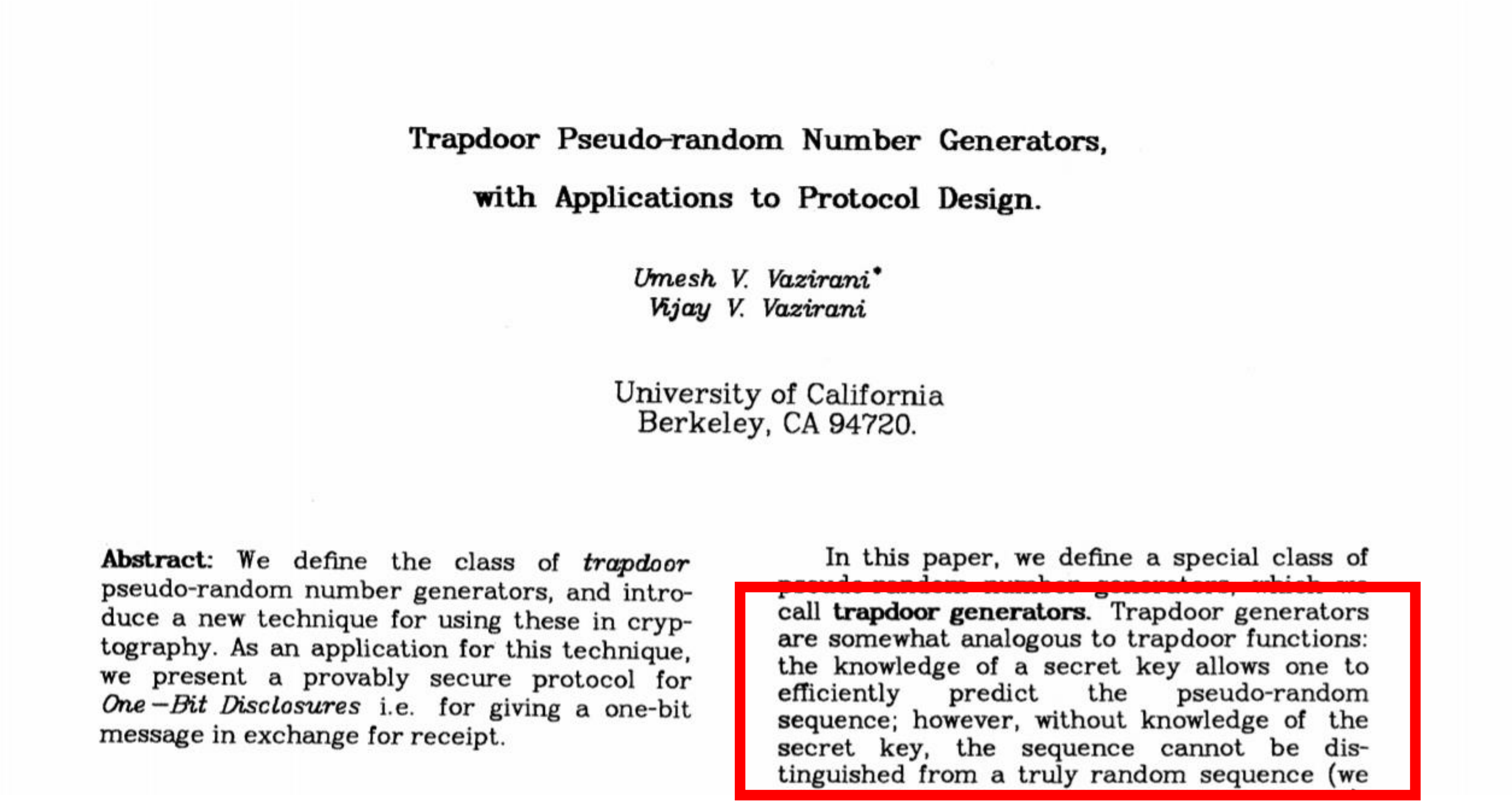
Umesh V. Vazirani[*]
Vijay V. Vazirani

University of California
Berkeley, CA 94720.

Abstract: We define the class of *trapdoor* pseudo-random number generators, and introduce a new technique for using these in cryptography. As an application for this technique, we present a provably secure protocol for *One–Bit Disclosures* i.e. for giving a one-bit message in exchange for receipt.

In this paper, we define a special class of pseudo-random number generators, which we call **trapdoor generators**. Trapdoor generators are somewhat analogous to trapdoor functions: the knowledge of a secret key allows one to efficiently predict the pseudo-random sequence; however, without knowledge of the secret key, the sequence cannot be distinguished from a truly random sequence (we

Umesh Vazirani, Vijay Vazirani FOCS 1983



Trapdoor Pseudo-random Number Generators,

with Applications to Protocol Design.

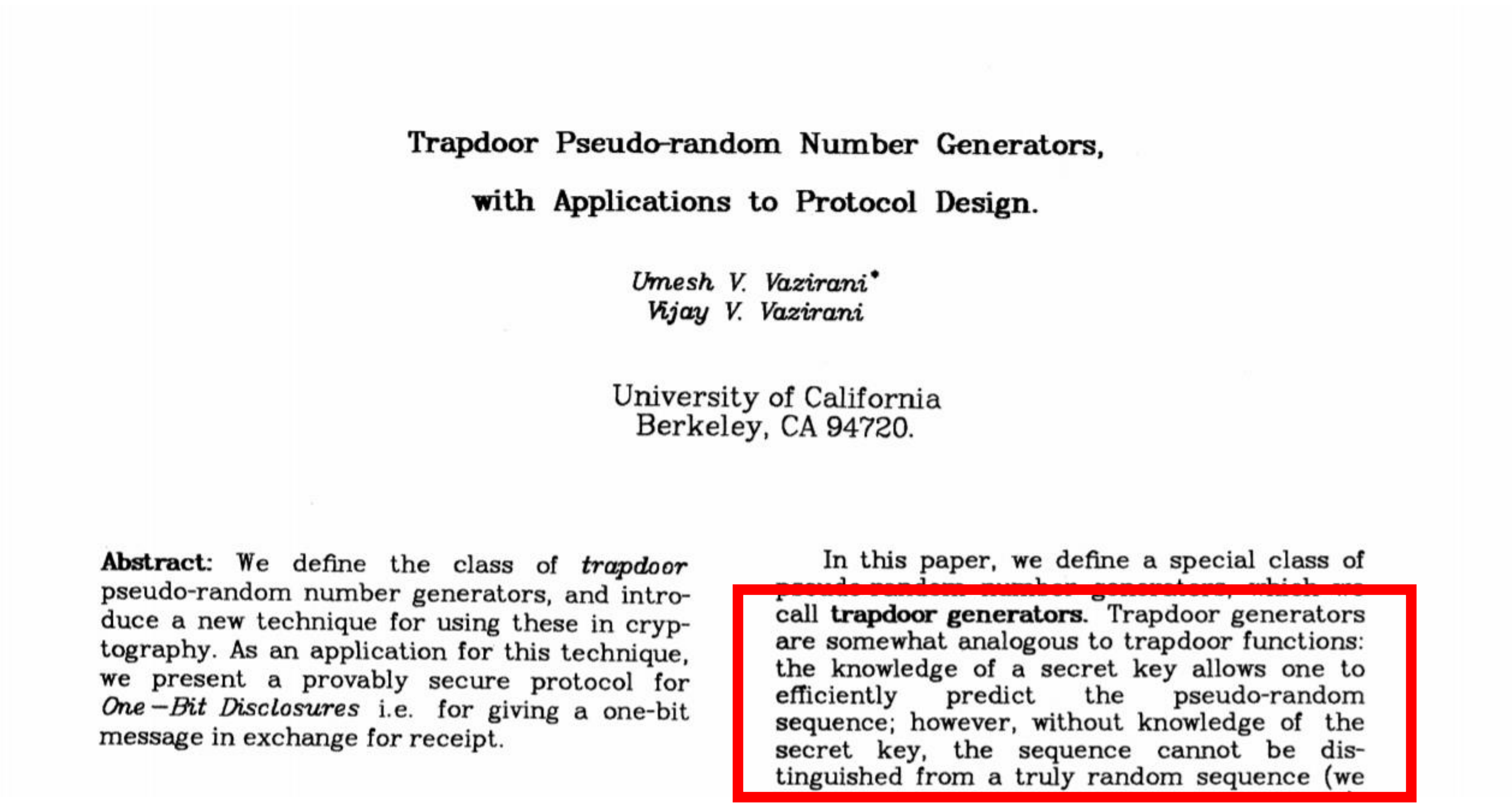
Umesh V. Vazirani[*]
Vijay V. Vazirani

University of California
Berkeley, CA 94720.

Abstract: We define the class of *trapdoor* pseudo-random number generators, and introduce a new technique for using these in cryptography. As an application for this technique, we present a provably secure protocol for *One – Bit Disclosures* i.e. for giving a one-bit message in exchange for receipt.

In this paper, we define a special class of pseudo-random number generators, which we call **trapdoor generators**. Trapdoor generators are somewhat analogous to trapdoor functions: the knowledge of a secret key allows one to efficiently predict the pseudo-random sequence; however, without knowledge of the secret key, the sequence cannot be distinguished from a truly random sequence (we

Umesh Vazirani, Vijay Vazirani FOCS 1983



Trapdoor Pseudo-random Number Generators, with Applications to Protocol Design.

*Umesh V. Vazirani**
Vijay V. Vazirani

University of California
Berkeley, CA 94720.

Abstract: We define the class of *trapdoor* pseudo-random number generators, and introduce a new technique for using these in cryptography. As an application for this technique, we present a provably secure protocol for *One-Bit Disclosures* i.e. for giving a one-bit message in exchange for receipt.

In this paper, we define a special class of pseudo-random number generators, which we call **trapdoor generators**. Trapdoor generators are somewhat analogous to trapdoor functions: the knowledge of a secret key allows one to efficiently predict the pseudo-random sequence; however, without knowledge of the secret key, the sequence cannot be distinguished from a truly random sequence (we

Used to construct protocols

A Formal Treatment of Backdoored Pseudorandom Generators

Yevgeniy Dodis[1], Chaya Ganesh[1], Alexander Golovnev[1], Ari Juels[2], and Thomas Ristenpart[3]

[1]Department of Computer Science, New York University
{dodis,ganesh,golovnev}@cs.nyu.edu
[2]Jacobs Institute, Cornell Tech, juels@cornell.edu
[3]Department of Computer Sciences, University of Wisconsin, rist@cs.wisc.edu

Pseudorandom Generators (PRGs)

$$pp \leftarrow setup$$

$$s_0 \leftarrow init(pp)$$

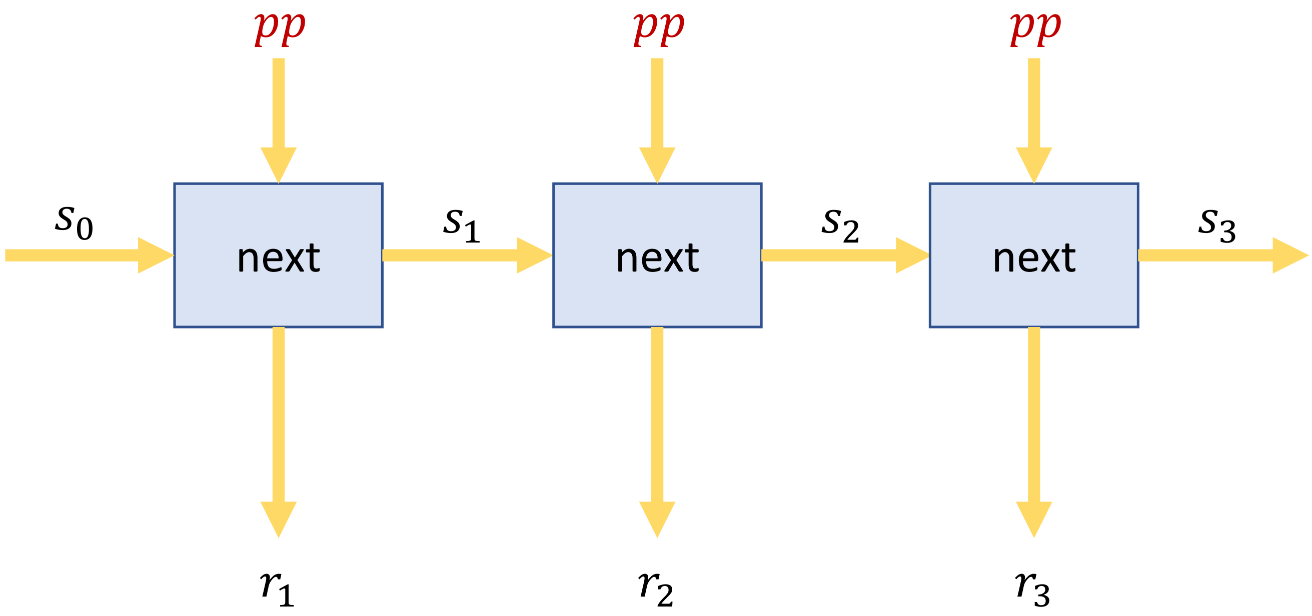
$$(s', r) \leftarrow next(pp, s)$$

Pseudorandom Generators (PRGs)

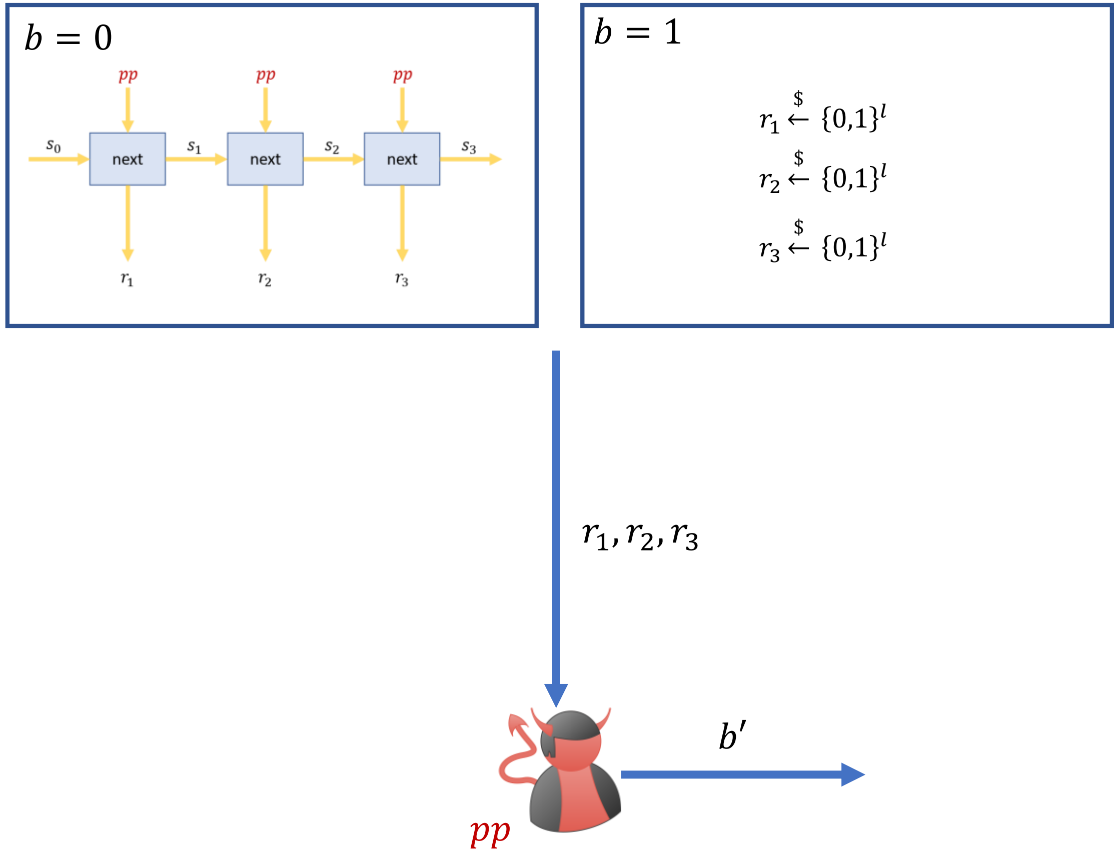
$pp \leftarrow setup$

$s_0 \leftarrow init(pp)$

$(s', r) \leftarrow next(pp, s)$



Security



$b = 0$

$b = 1$

$r_1 \overset{\$}{\leftarrow} \{0,1\}^l$

$r_2 \overset{\$}{\leftarrow} \{0,1\}^l$

$r_3 \overset{\$}{\leftarrow} \{0,1\}^l$

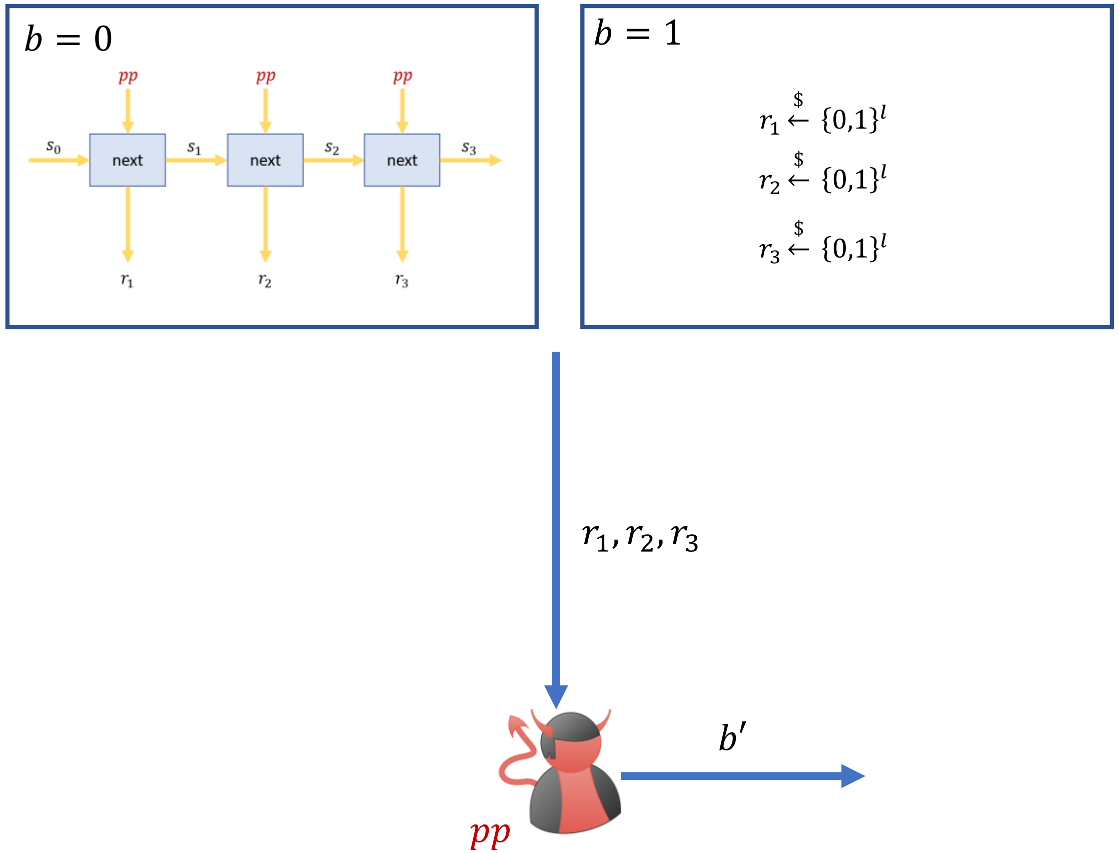
r_1, r_2, r_3

b'

pp

distinguishing security

Security



$b = 0$

pp pp pp

s_0 next s_1 next s_2 next s_3

r_1 r_2 r_3

$b = 1$

$r_1 \overset{\$}{\leftarrow} \{0,1\}^l$

$r_2 \overset{\$}{\leftarrow} \{0,1\}^l$

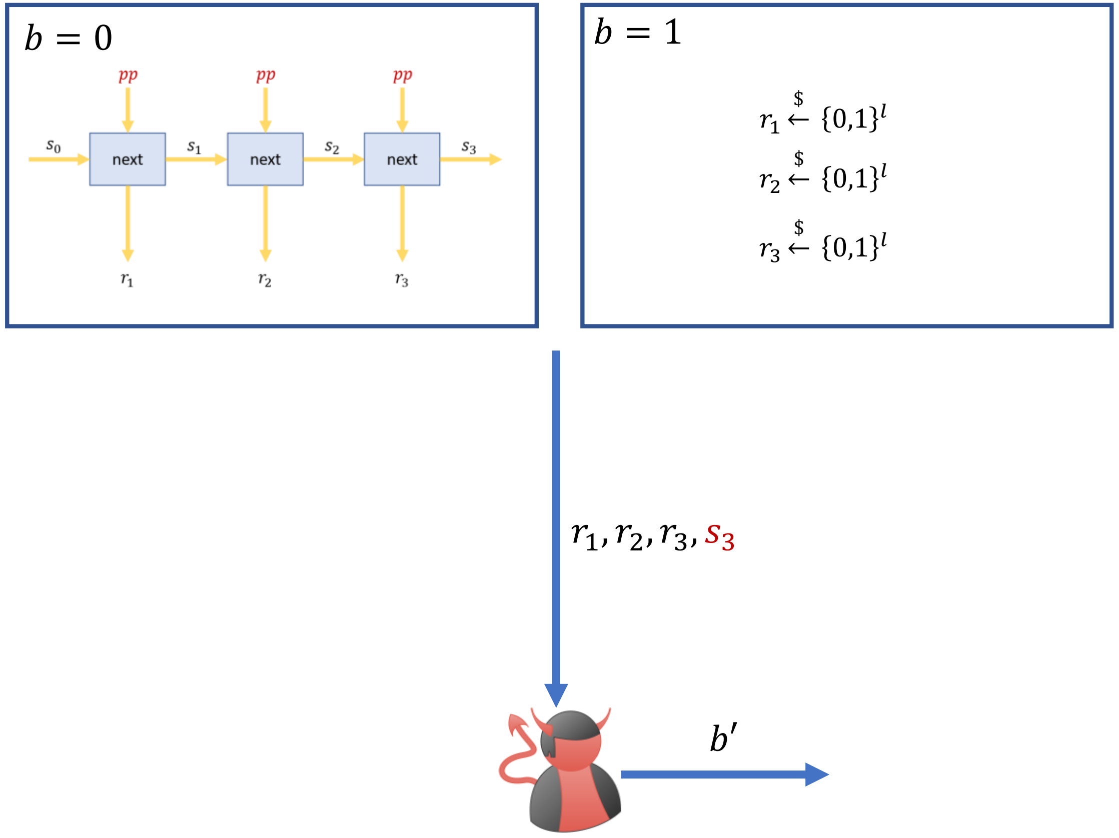
$r_3 \overset{\$}{\leftarrow} \{0,1\}^l$

r_1, r_2, r_3

b'

pp

distinguishing security



$b = 0$

pp pp pp

s_0 next s_1 next s_2 next s_3

r_1 r_2 r_3

$b = 1$

$r_1 \overset{\$}{\leftarrow} \{0,1\}^l$

$r_2 \overset{\$}{\leftarrow} \{0,1\}^l$

$r_3 \overset{\$}{\leftarrow} \{0,1\}^l$

r_1, r_2, r_3, s_3

b'

pp

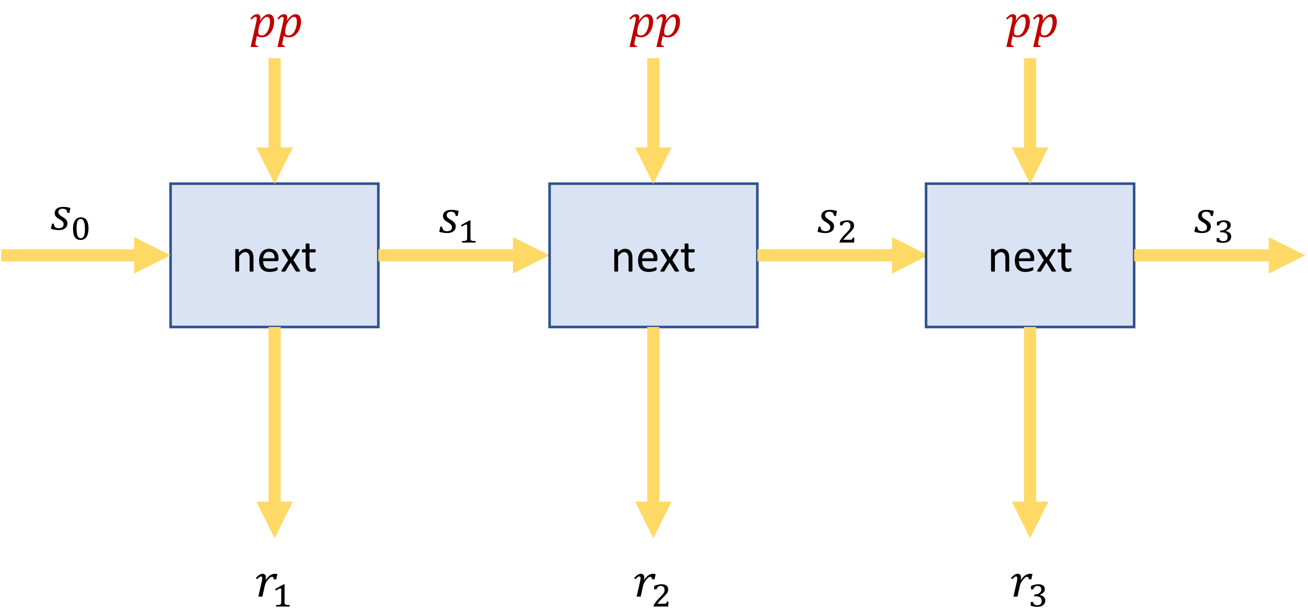
forward security

Backdoored PRGs

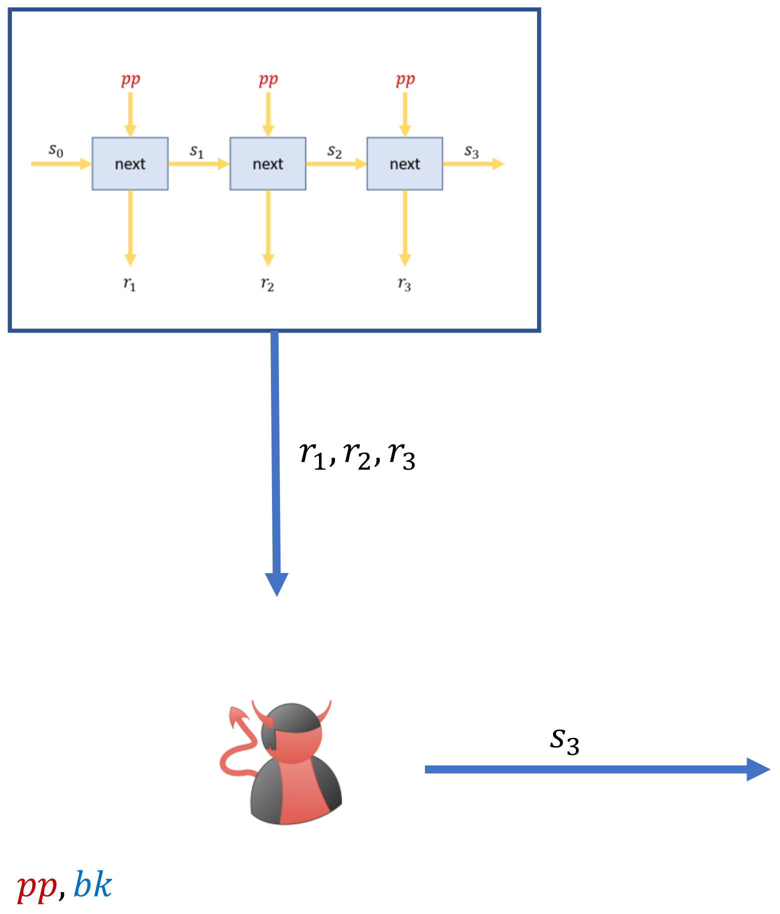
$pp, bk \leftarrow setup$

$s_0 \leftarrow init(pp)$

$(s', r) \leftarrow next(pp, s)$



Backdooring



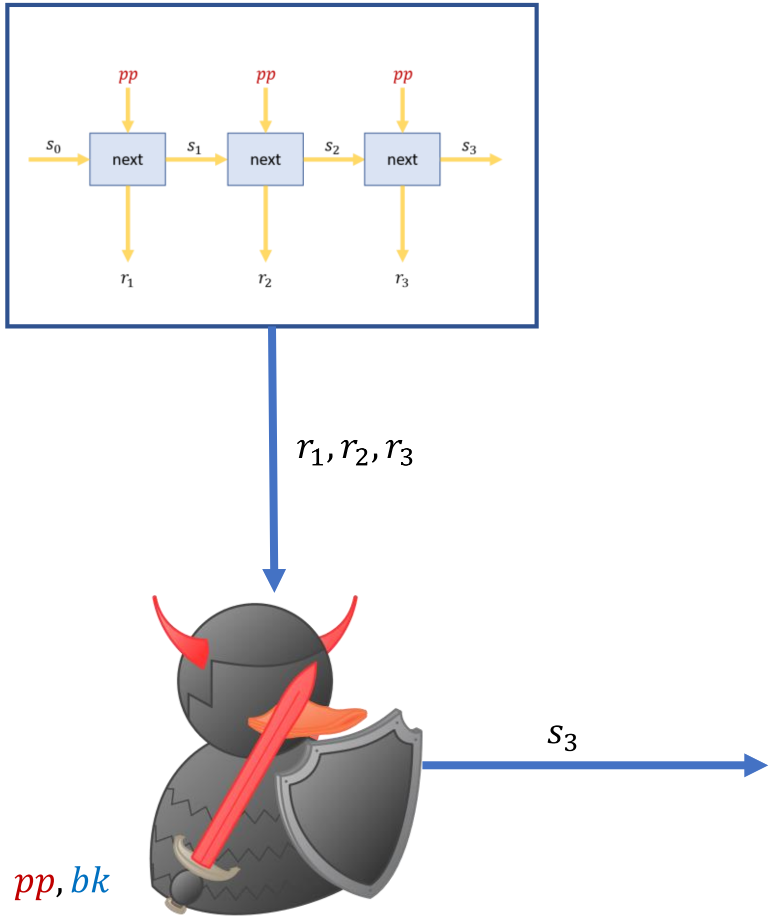
r_1, r_2, r_3

s_3

pp, bk

current state recovery

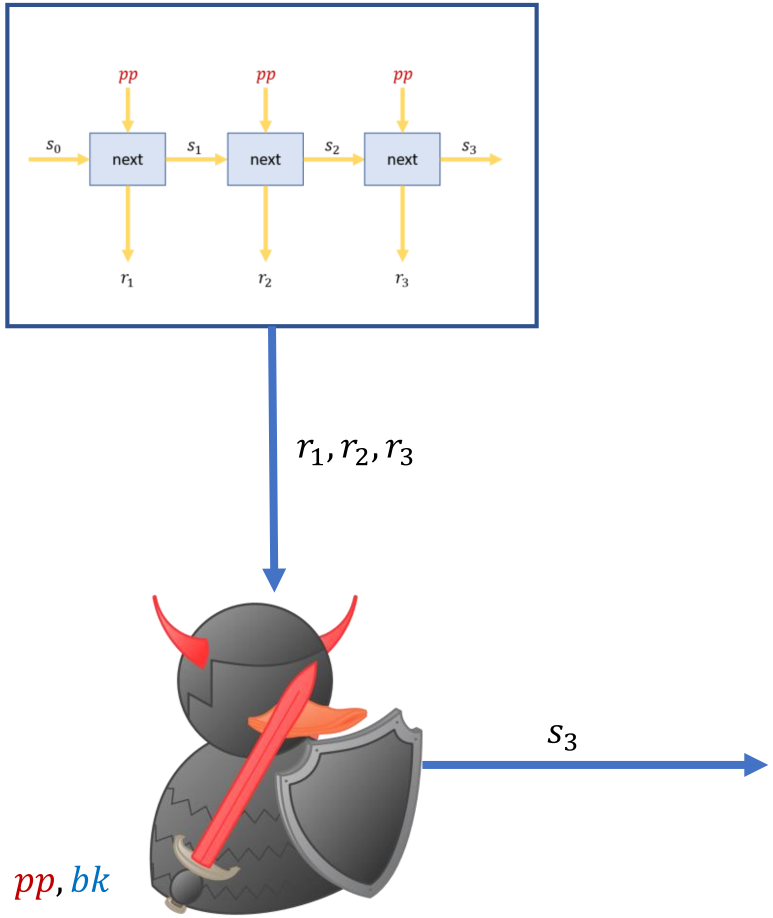
Backdooring



current state recovery

Backdooring

BPRG if

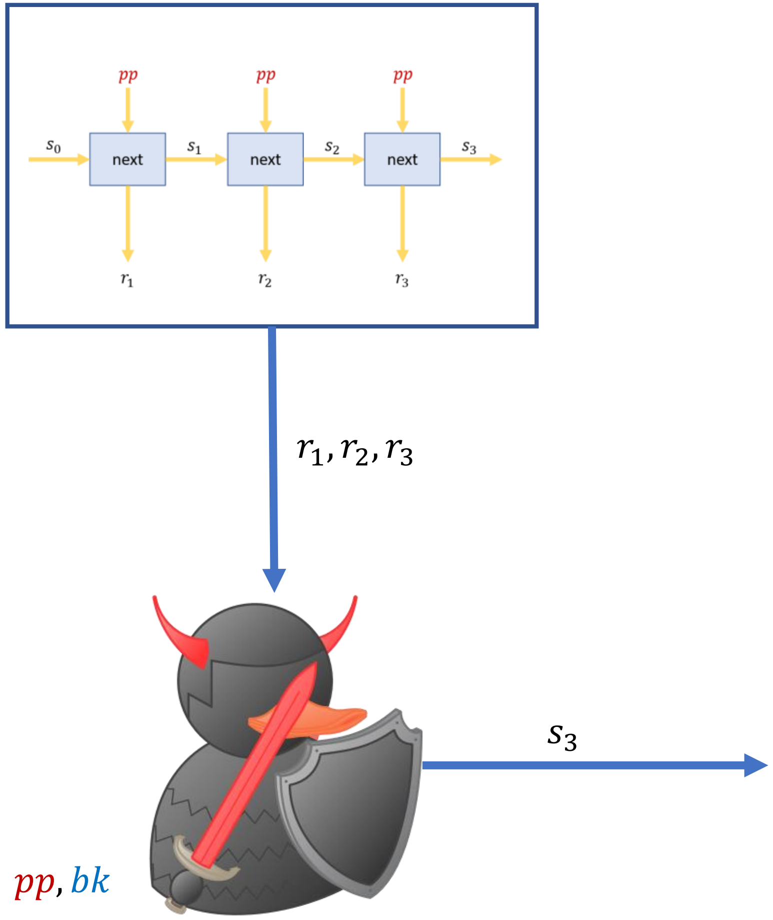


current state recovery

Backdooring

BPRG if

1. PRG secure against all

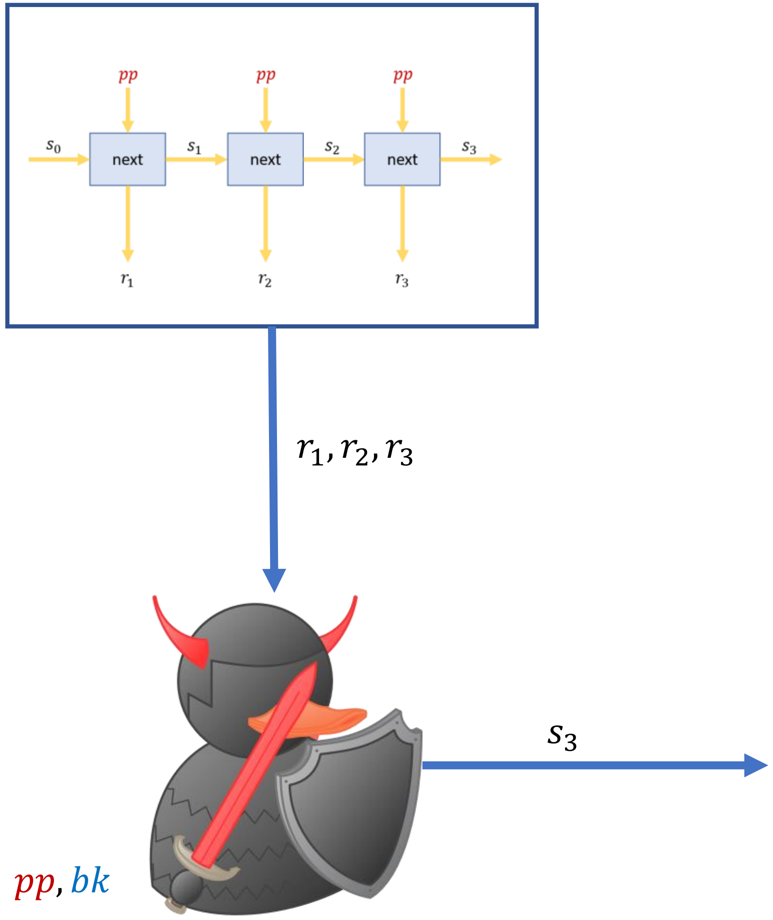


current state recovery

Backdooring

BPRG if

1. PRG secure against all



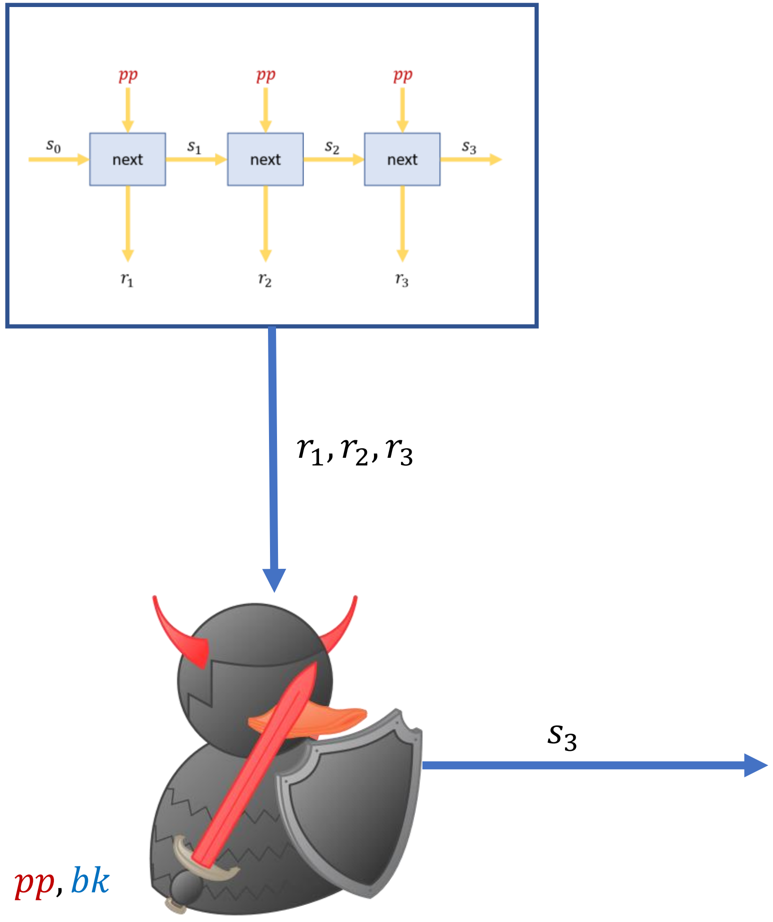
current state recovery

Backdooring

BPRG if

1. PRG secure against all

2. State recovery successful by

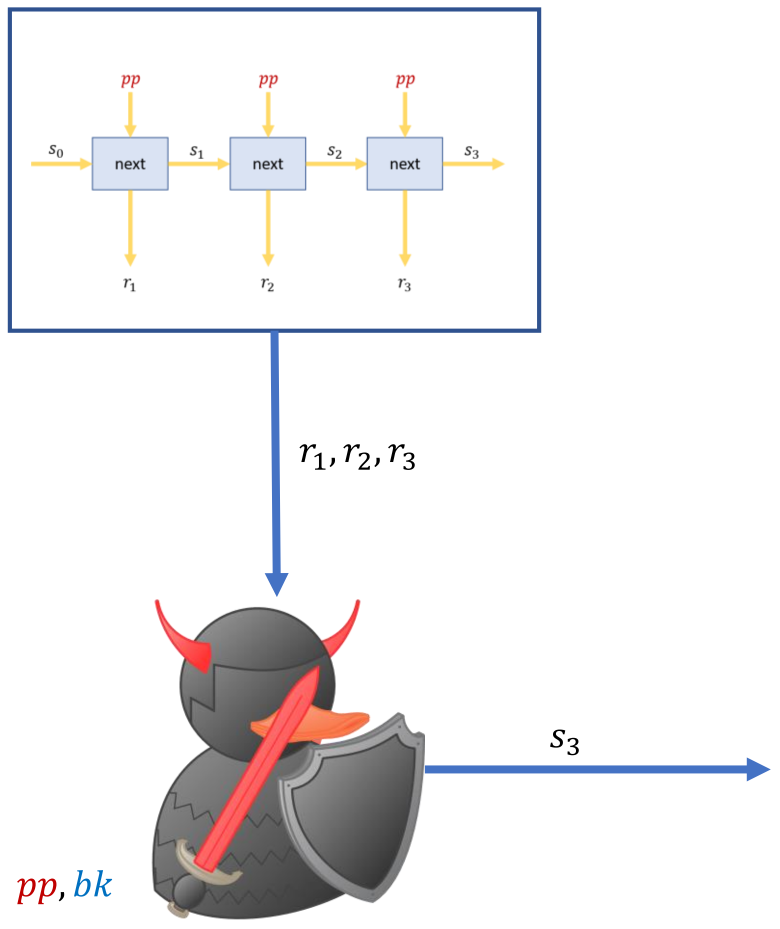


current state recovery

Backdooring

BPRG if

1. PRG secure against all
2. State recovery successful by



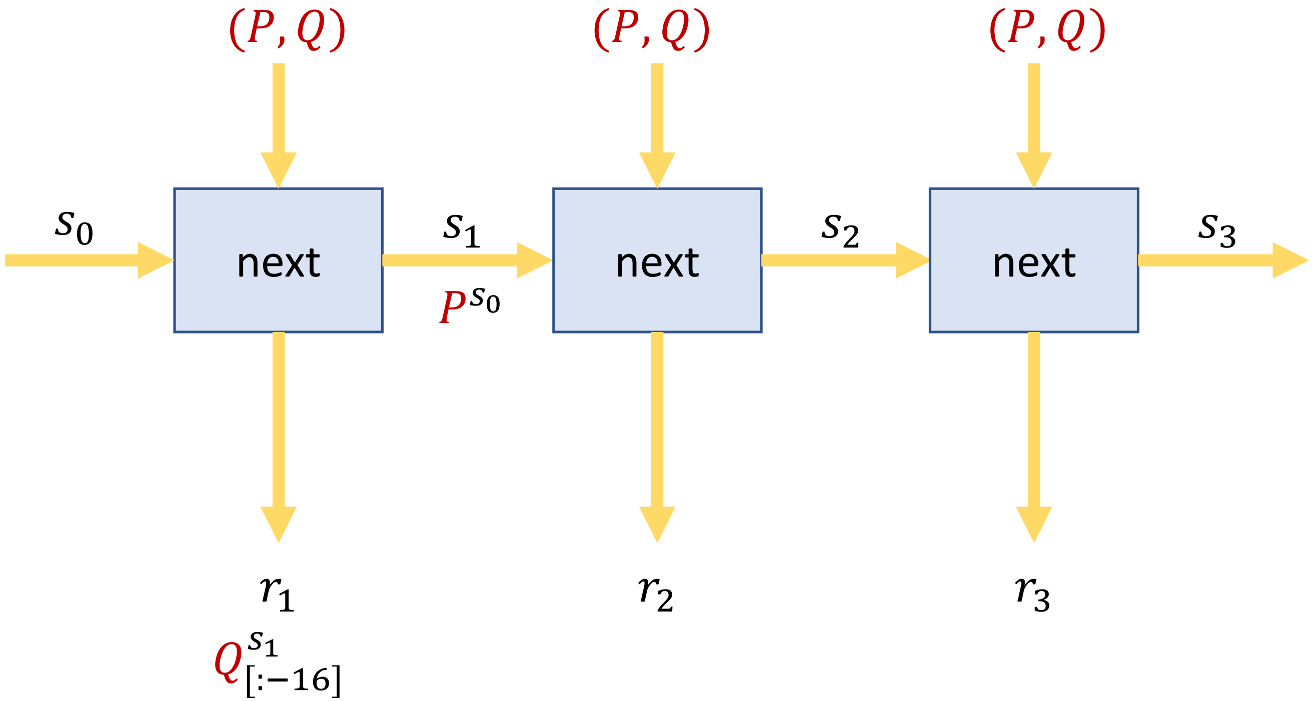
current state recovery

Dual EC DRBG

$(P, Q), d \leftarrow setup$ $\quad P = Q^d$

$s_0 \leftarrow init(pp)$

$\left(P^s, Q^{s'}_{[:-16]}\right) \leftarrow next((P, Q), s)$

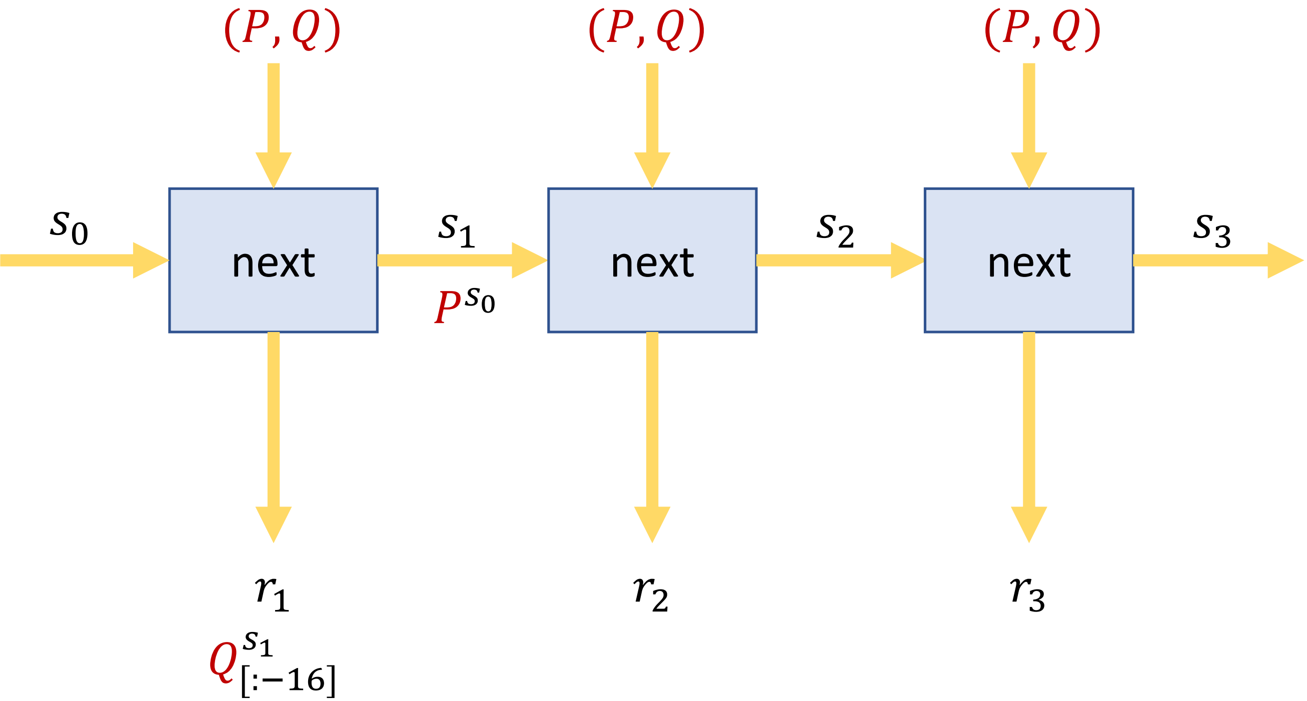


Dual EC DRBG

$(P, Q), d \leftarrow setup$ $\quad P = Q^d$

$s_0 \leftarrow init(pp)$

$\left(P^s, Q^{s'}_{[:-16]}\right) \leftarrow next((P, Q), s)$



$(P, Q), d$

Dual EC DRBG

How are P and Q generated in practice?

Dual EC DRBG

How are P and Q generated in practice?

What about in the NIST standard?

Dual EC DRBG

How are P and Q generated in practice?

What about in the NIST standard?

Does not produce output provably indistinguishable from random.

Dual EC DRBG

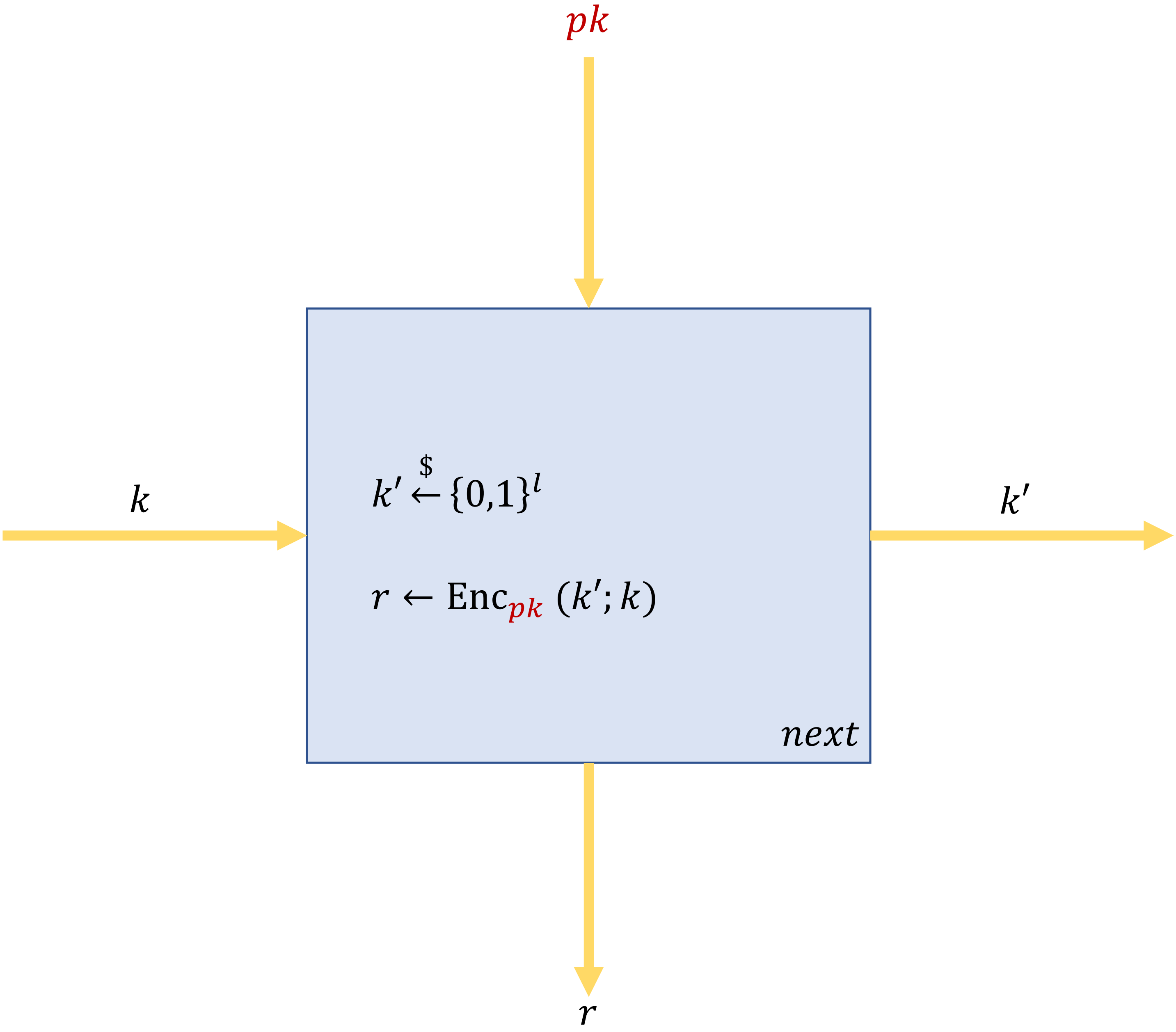
How are P and Q generated in practice?

What about in the NIST standard?

Does not produce output provably indistinguishable from random.

Can we build a backdoored PRG?

Construction



pk

k

$k' \overset{\$}{\leftarrow} \{0,1\}^l$

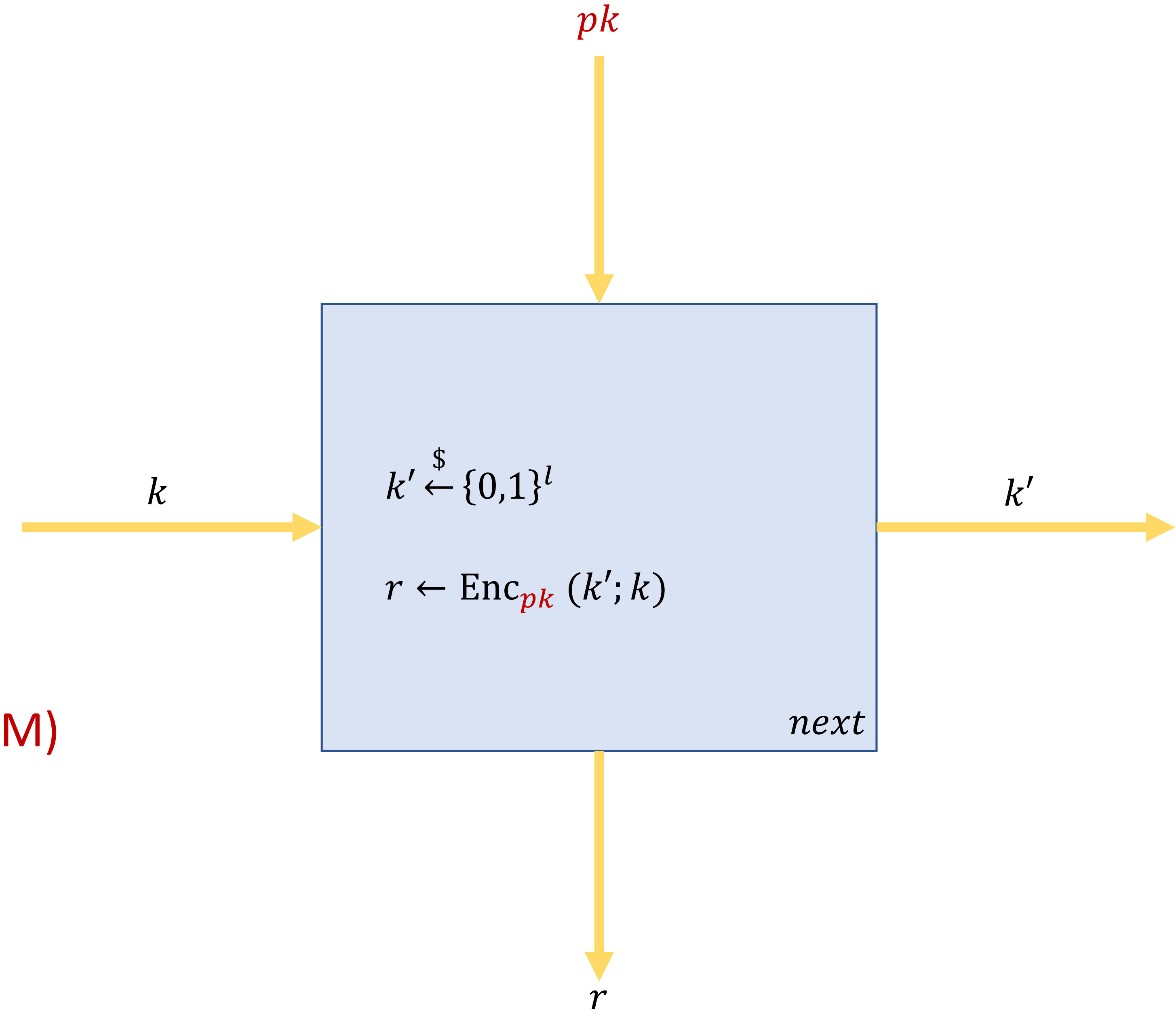
$r \leftarrow \mathrm{Enc}_{pk}\,(k'; k)$

k'

$next$

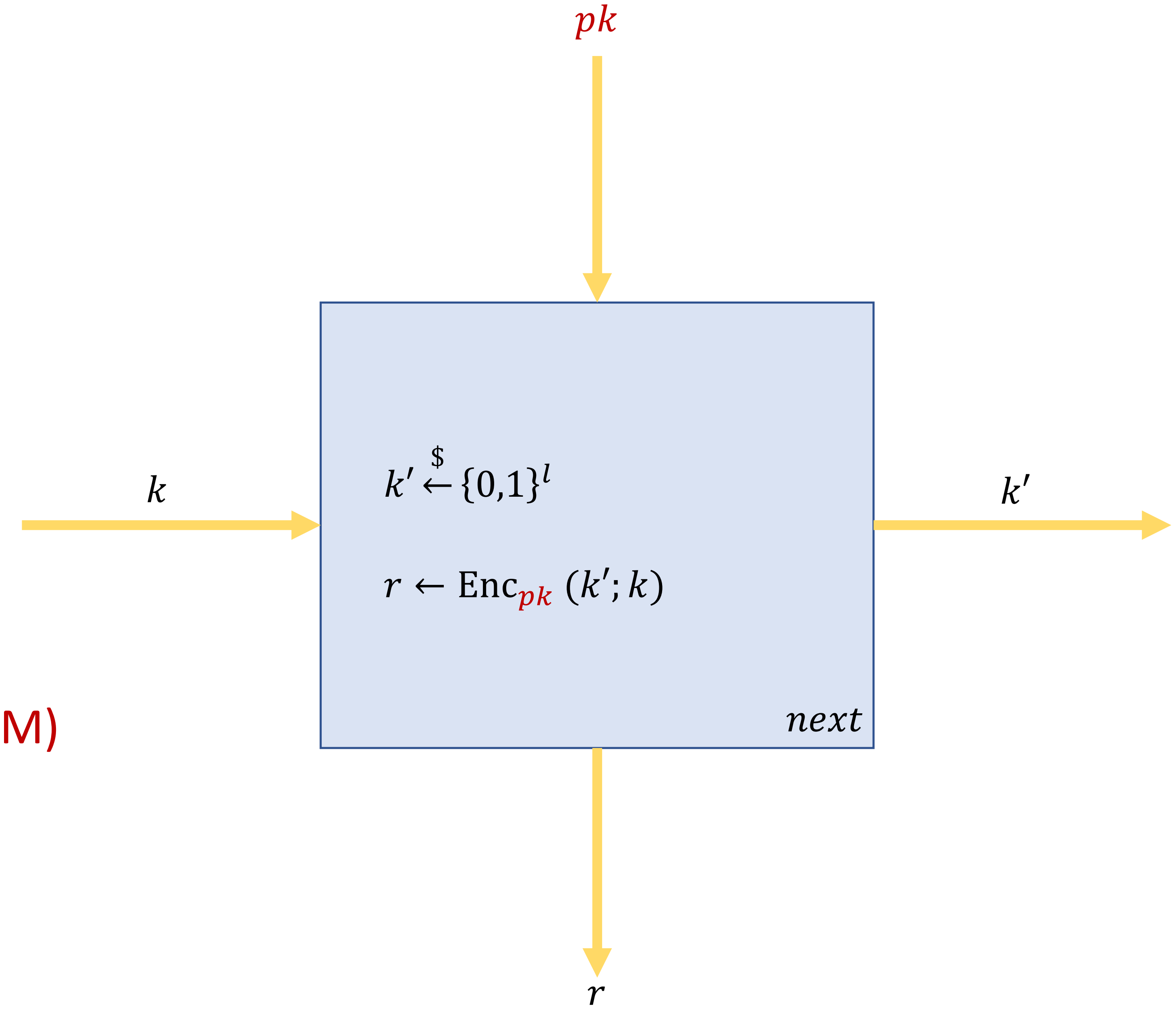
r

Construction


pk
k
$k' \xleftarrow{\$} \{0,1\}^l$
$r \leftarrow \mathrm{Enc}_{pk}\,(k'; k)$
k'
$next$
r

Key encapsulation mechanism (KEM)

Construction



pk

k

$k' \xleftarrow{\$} \{0,1\}^l$

$r \leftarrow \mathrm{Enc}_{pk}\ (k'; k)$

k'

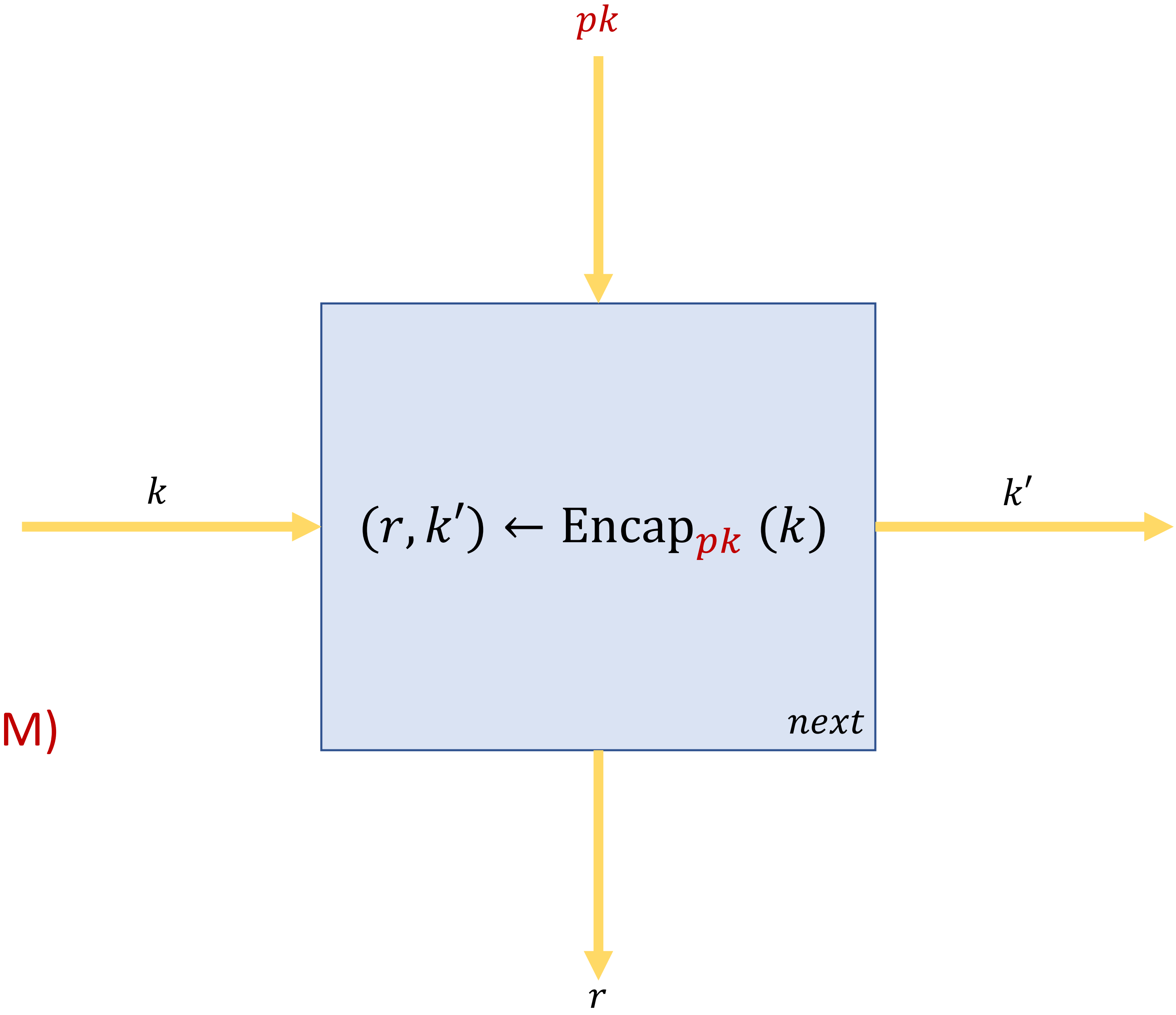
$next$

r

Key encapsulation mechanism (KEM)

Pseudorandom ciphertexts

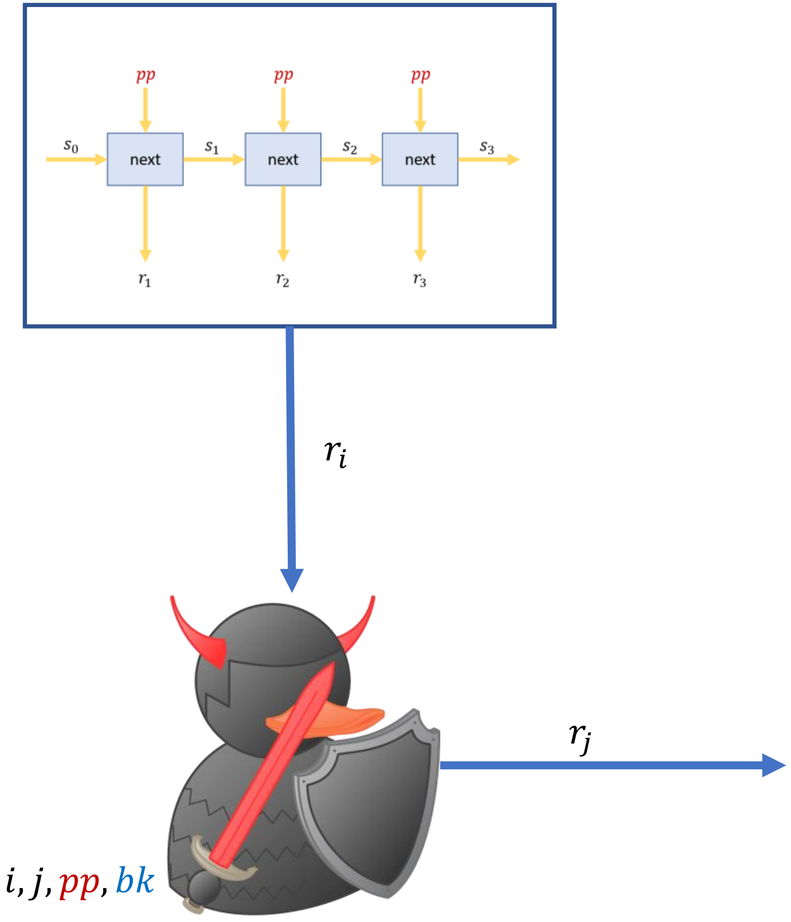
Construction



Key encapsulation mechanism (KEM)

Pseudorandom ciphertexts

Stronger Backdooring



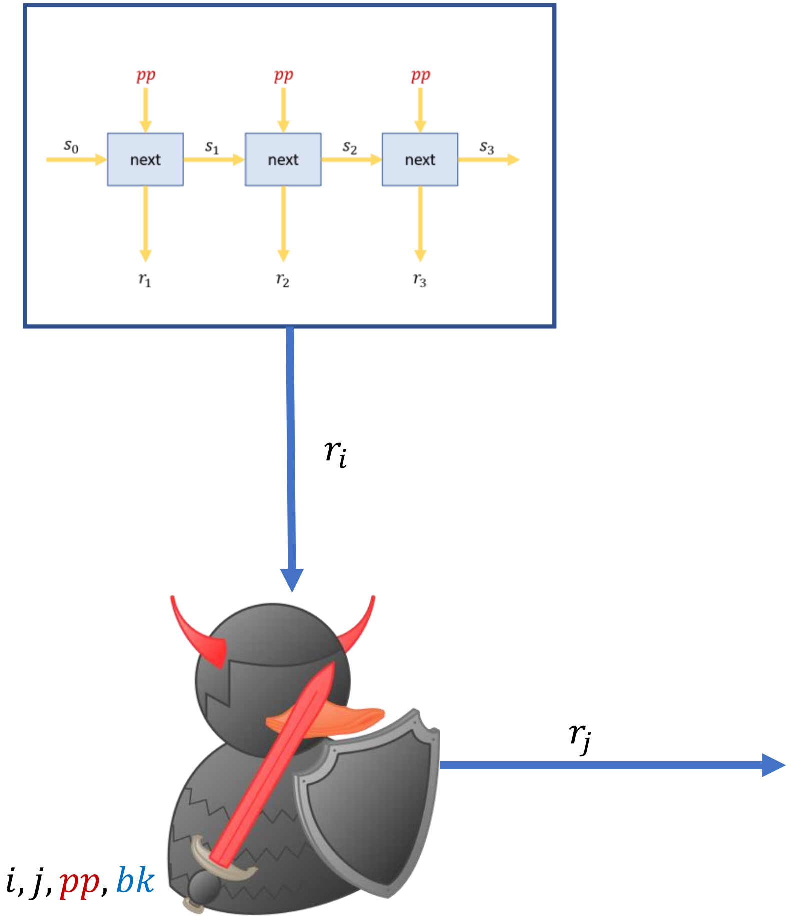
random seek

Stronger Backdooring

BPRG if

1. PRG secure against all

2. Random seek successful by

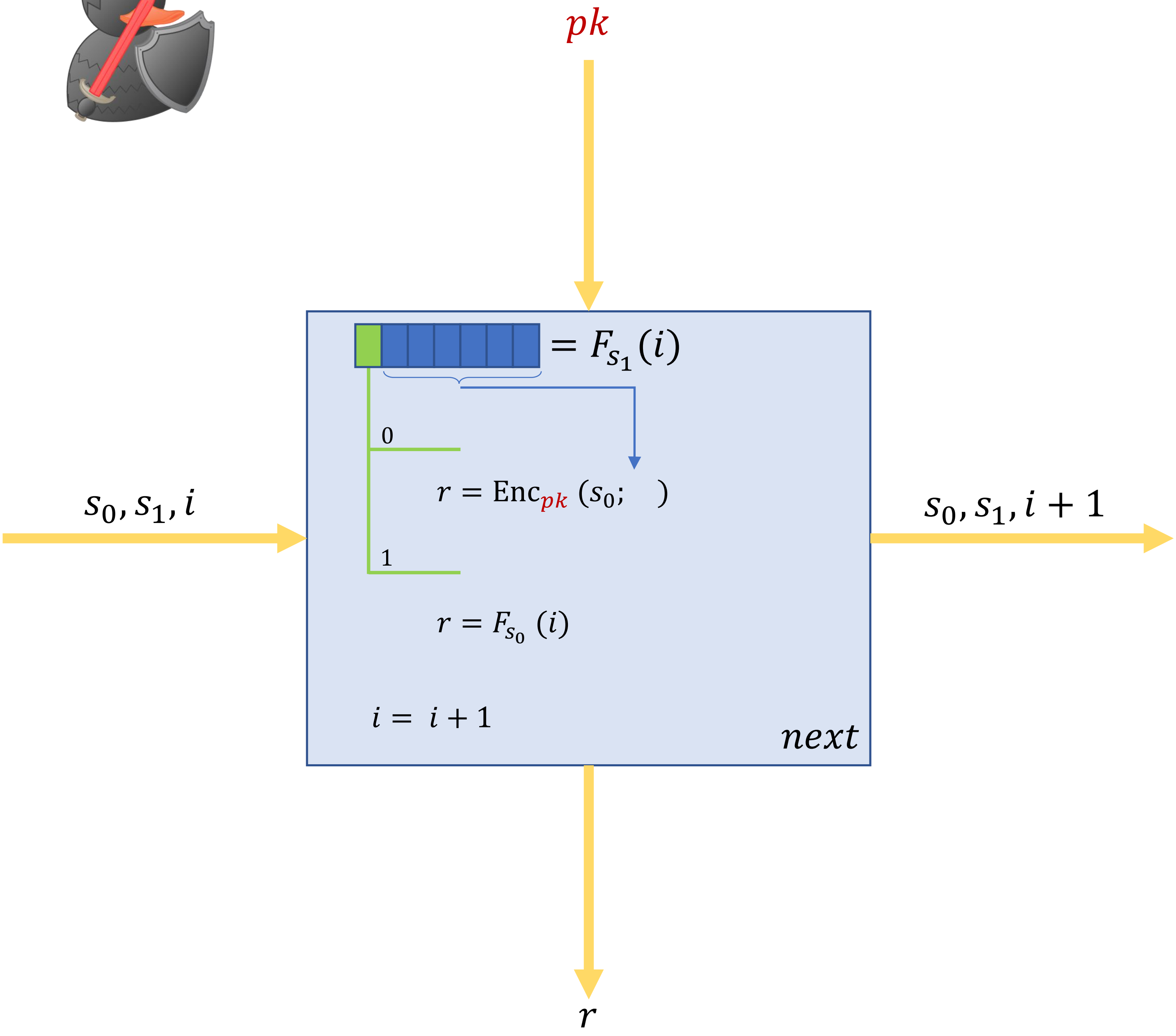


random seek

Backdoored PRGs

$$\underbrace{{\color{red}pk}, {\color{blue}sk}}_{\text{public key encryption with pseudorandom ciphertexts}} \leftarrow setup$$

$$(\underbrace{s_0, s_1}_{\text{keys of a PRF}}, 0) \leftarrow init({\color{red}pk})$$



pk

s_0, s_1, i

$= F_{s_1}(i)$

0

$r = \mathrm{Enc}_{pk}(s_0;\quad)$

1

$r = F_{s_0}(i)$

$i = i + 1$

$next$

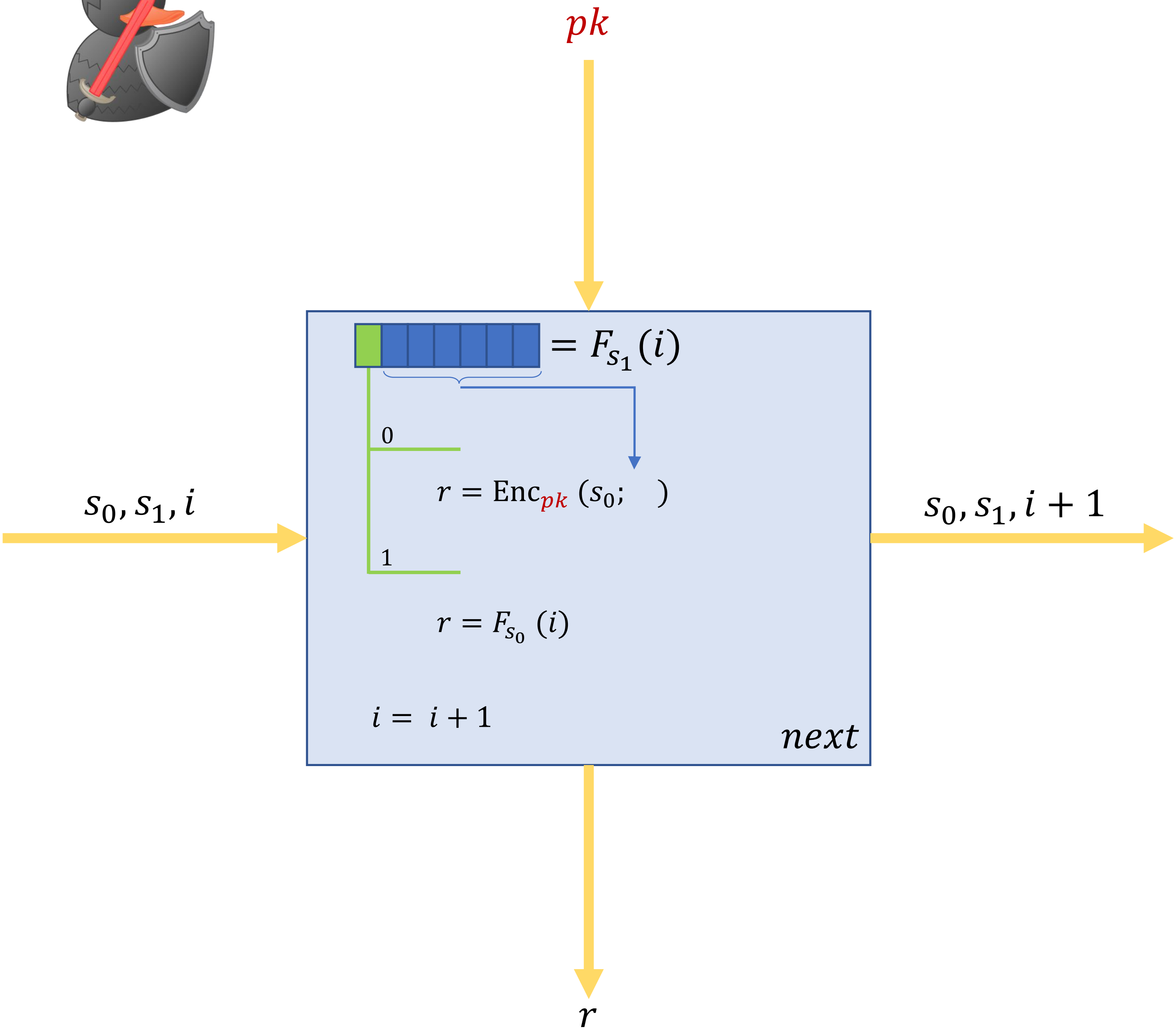
$s_0, s_1, i + 1$

r

Backdoored PRGs

$$\underbrace{pk, sk}_{\text{public key encryption with pseudorandom ciphertexts}} \leftarrow setup$$

$$(\underbrace{s_0, s_1}_{\text{keys of a PRF}}, 0) \leftarrow init(pk)$$



Forward secure?

Equivalence

Public-encryption with pseudorandom ciphertexts

$\Longleftrightarrow$

Backdoor PRG

Equivalence

Public-encryption with pseudorandom ciphertexts

$\Longleftrightarrow$

Backdoor PRG

PRGs built from symmetric key primitives unlikely to be backdoored

Counter Measures

Counter Measures

Don’t use non-standard PRGs

- Not always possible

Counter Measures

Don’t use non-standard PRGs

- Not always possible

Post processing of output: Immunization

Immunization

If the saboteur knows the immunizer strategy in advance

Immunization

If the saboteur knows the immunizer strategy in advance

- Leak information of the initial state 1 bit at a time by rejection sampling
- Even if you use a hash function modeled as a random oracle

Immunization

If the saboteur knows the immunizer strategy in advance

- Leak information of the initial state 1 bit at a time by rejection sampling
- Even if you use a hash function modeled as a random oracle

If the immunization uses randomness not revealed to the saboteur

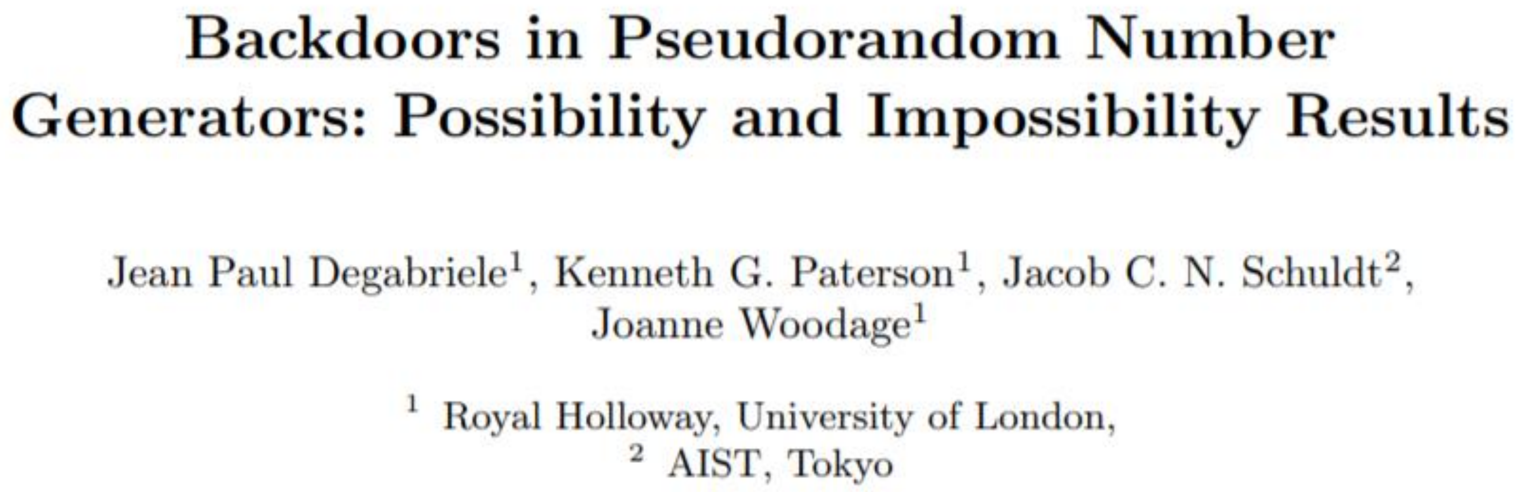
Immunization

If the saboteur knows the immunizer strategy in advance

- Leak information of the initial state 1 bit at a time by rejection sampling
- Even if you use a hash function modeled as a random oracle

If the immunization uses randomness not revealed to the saboteur

- Can be done, but not with trivial functions



Backdoors in Pseudorandom Number Generators: Possibility and Impossibility Results

Jean Paul Degabriele[1], Kenneth G. Paterson[1], Jacob C. N. Schuldt[2], Joanne Woodage[1]

[1] Royal Holloway, University of London,
[2] AIST, Tokyo

Backdooring PRGs

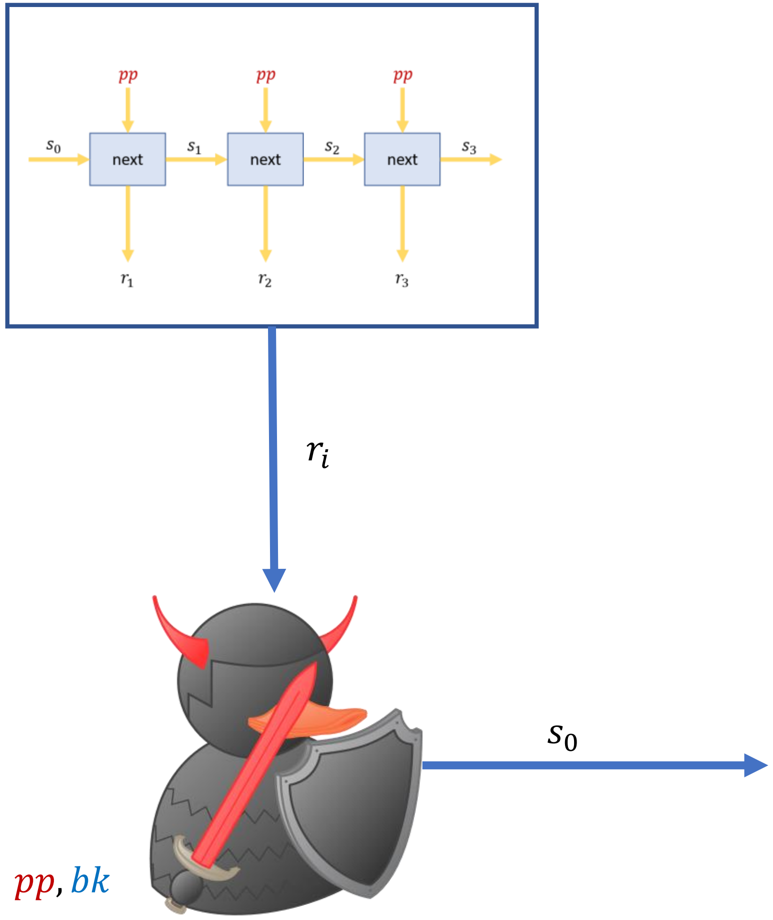
Can we hope to retrieve older states of the PRG while maintaining forward secrecy?

Backdooring PRGs

Can we hope to retrieve older states of the PRG while maintaining forward secrecy?

PRGs can be backdoored in the worst possible sense

Strongest Backdooring



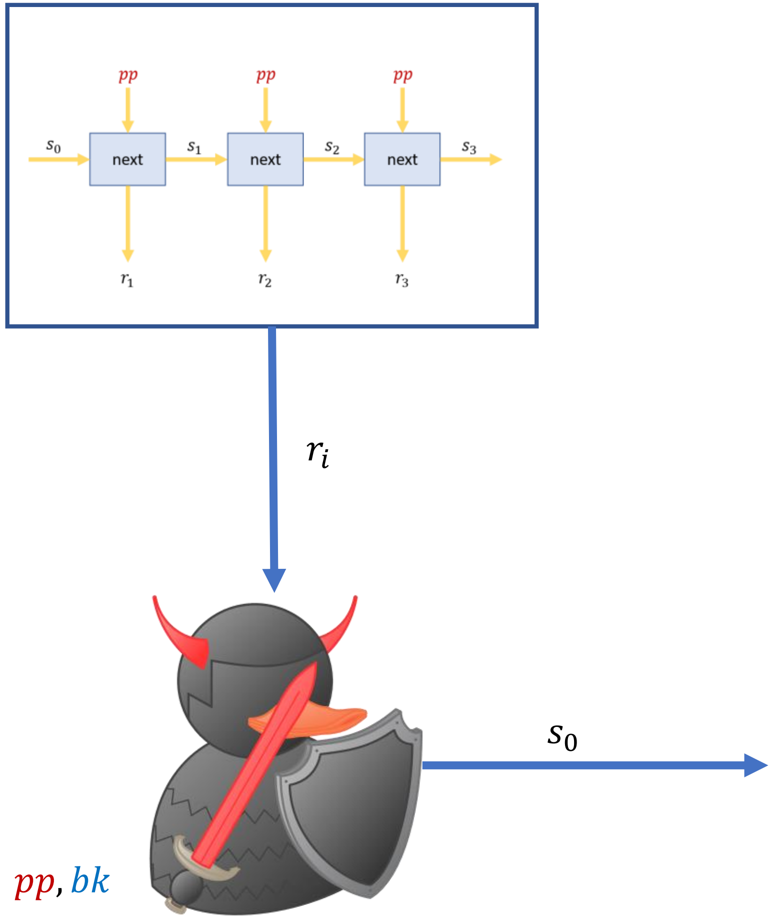
initial state recovery

Strongest Backdooring

BPRG if

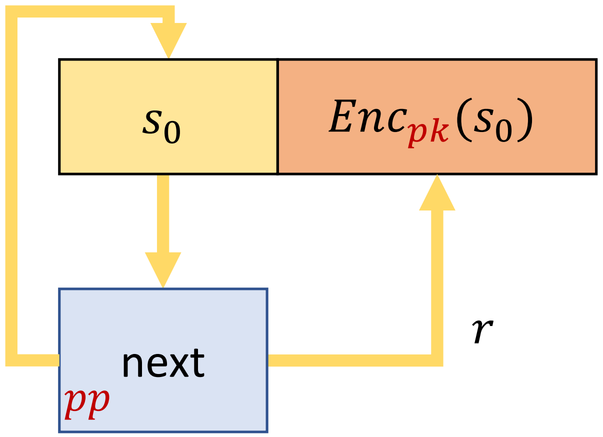
1. PRG secure against all

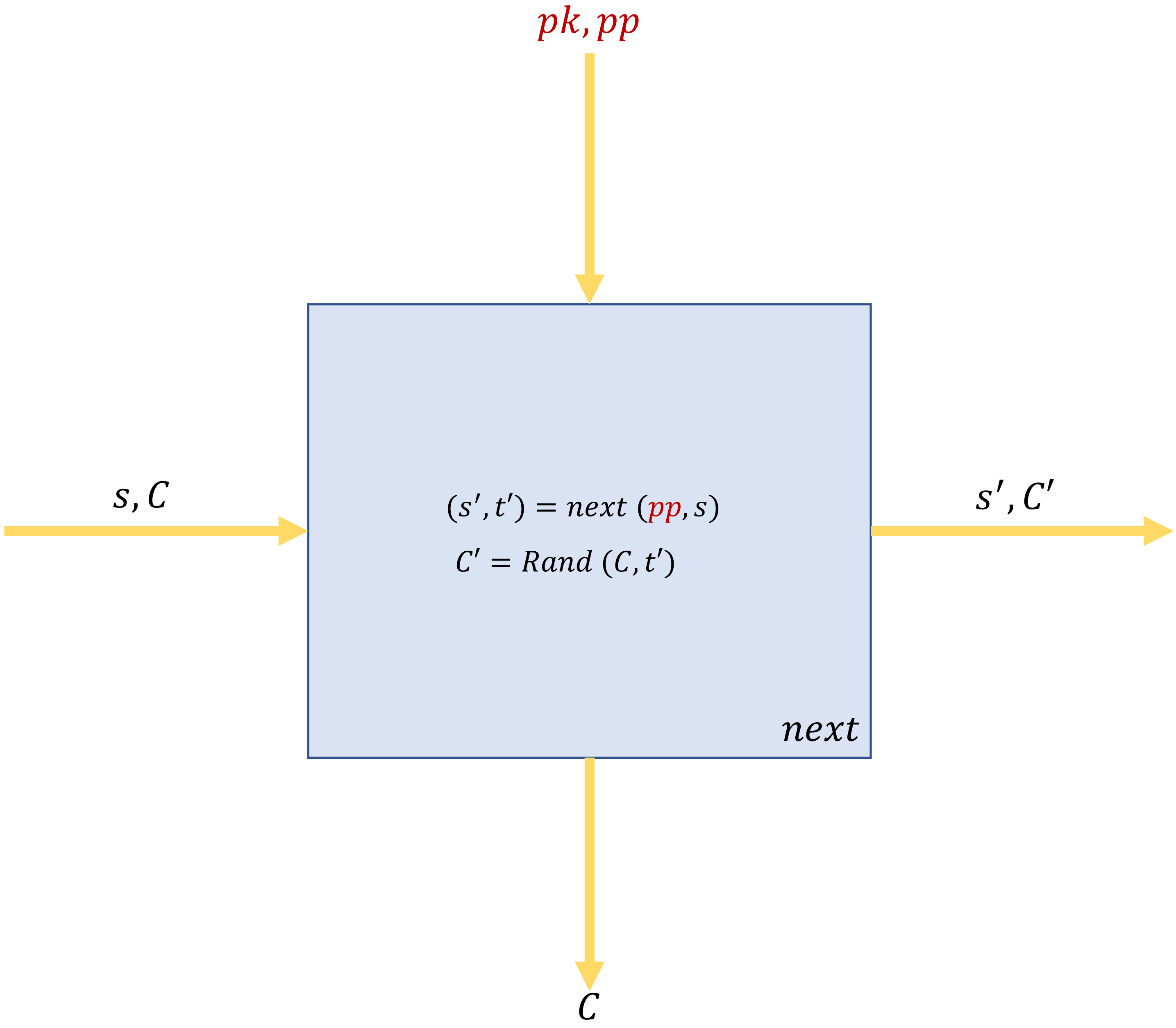
2. Initial state recovery successful by



initial state recovery

Construction





$(s', t') = next\ (pp, s)$

$C' = Rand\ (C, t')$

PRNGs with inputs

Only assumes secret state and access to some (potentially biased) random source.

PRNGs with inputs

Only assumes secret state and access to some (potentially biased) random source.

Preserve security when its entropy inputs are influenced by an attacker and to recover security after its state is compromised, via refreshing.

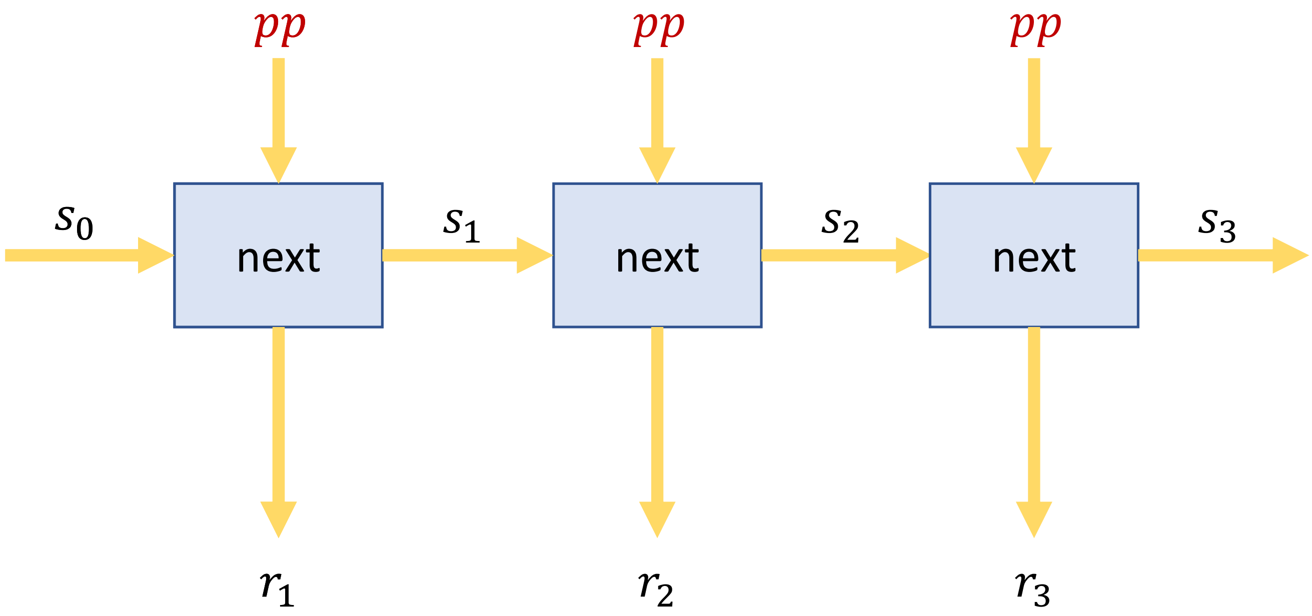
PRNG with Inputs

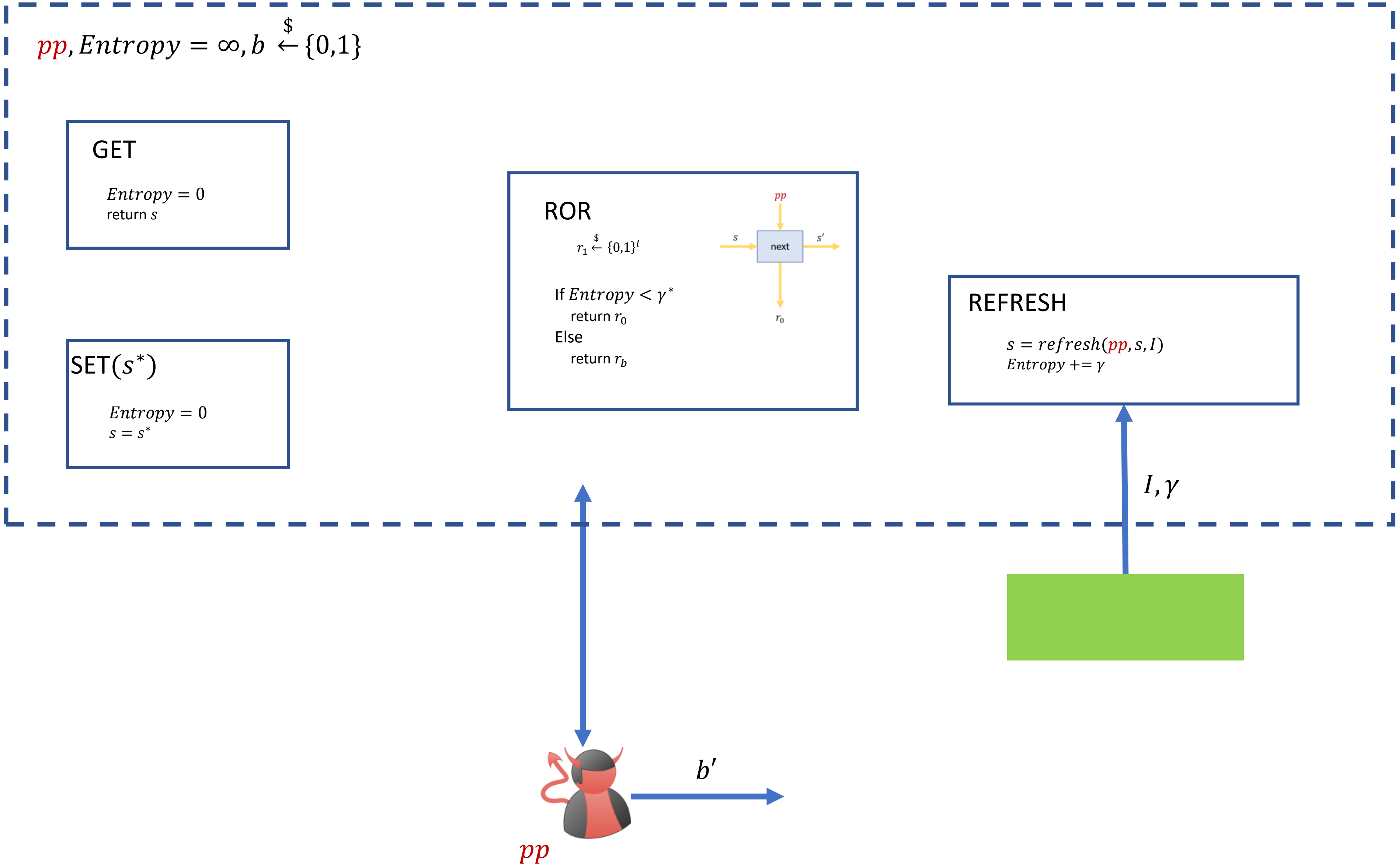
$pp \leftarrow setup$

$s_0 \leftarrow init(pp)$

$(s', r) \leftarrow next(pp, s)$

$s' \leftarrow refresh(s, I)$





$pp, Entropy = \infty, b \overset{\$}{\leftarrow} \{0,1\}$

GET

```
$Entropy = 0$
return $s$
```

SET(s^*)

```
$Entropy = 0$
$s = s^*$
```

ROR

```
$r_1 \overset{\$}{\leftarrow} \{0,1\}^l$

If $Entropy < \gamma^*$
    return $r_0$
Else
    return $r_b$
```

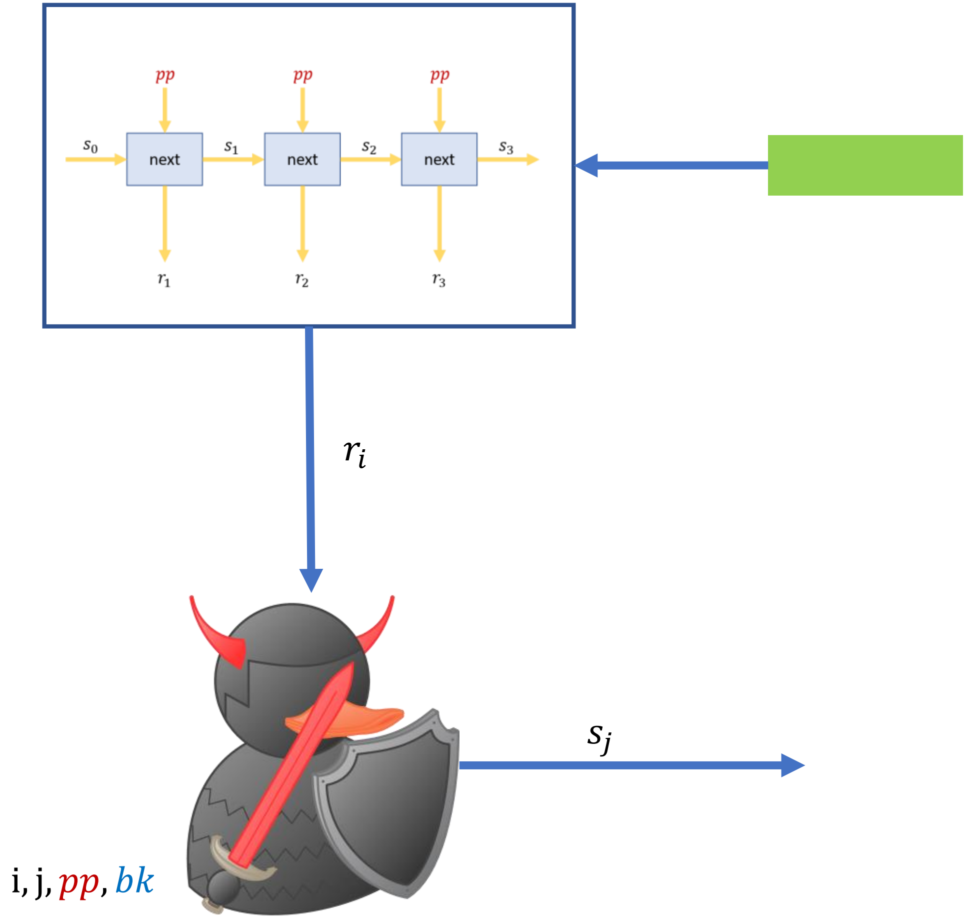
REFRESH

```
$s = refresh(pp, s, I)$
$Entropy \mathrel{+}= \gamma$
```

robustness

Backdooring PRNG

Needs to work for any sequence of $next$ and $refresh$ calls to the state.



Also given the sequence of calls to $next$ and $refresh$

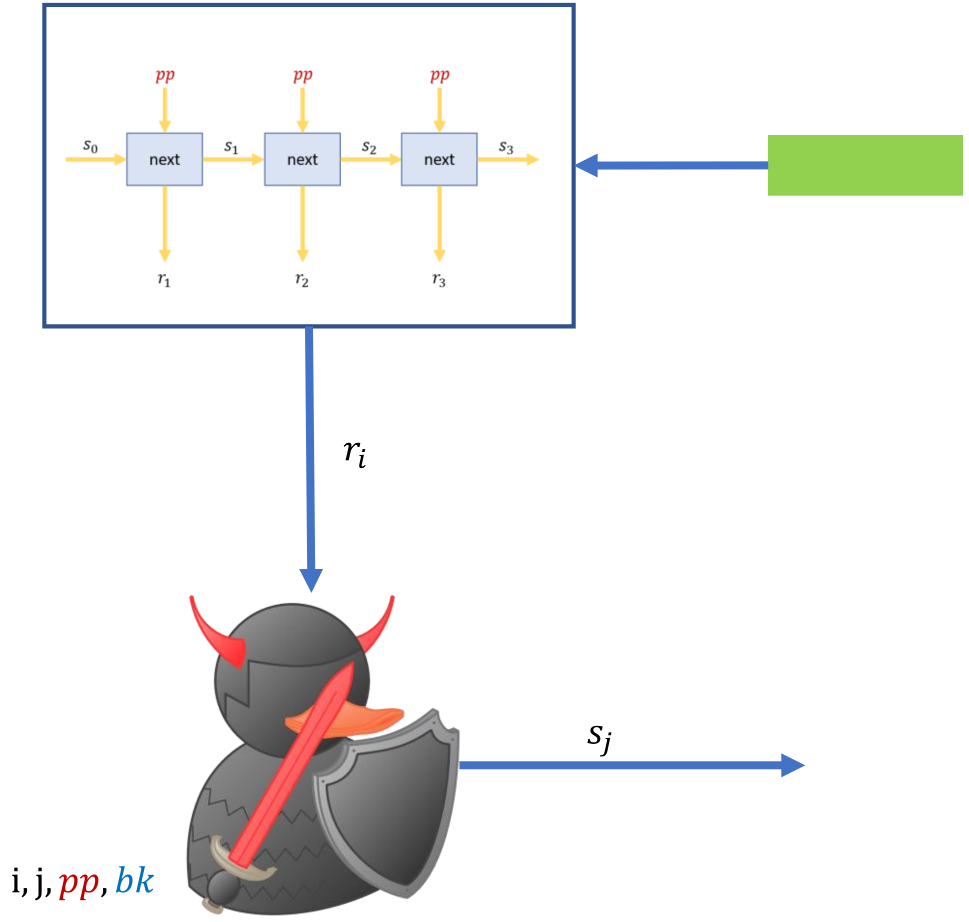
state recovery

Backdooring PRNG

Needs to work for any sequence of $next$ and $refresh$ calls to the state.

BPRG if

1. PRNG robust against all
2. State recovery successful by

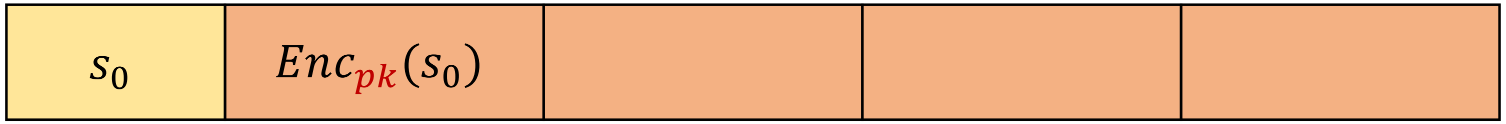


Also given the sequence of calls to $next$ and $refresh$

state recovery

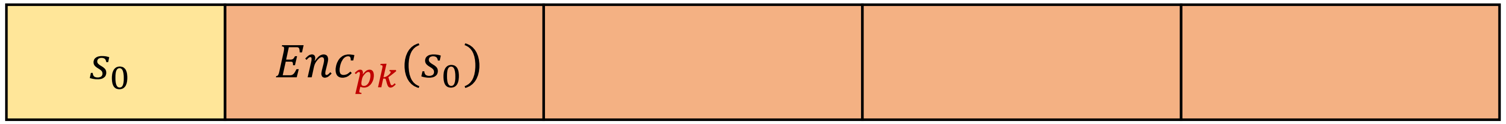
Overview of Construction

Keep snapshots of the state



Overview of Construction

Keep snapshots of the state



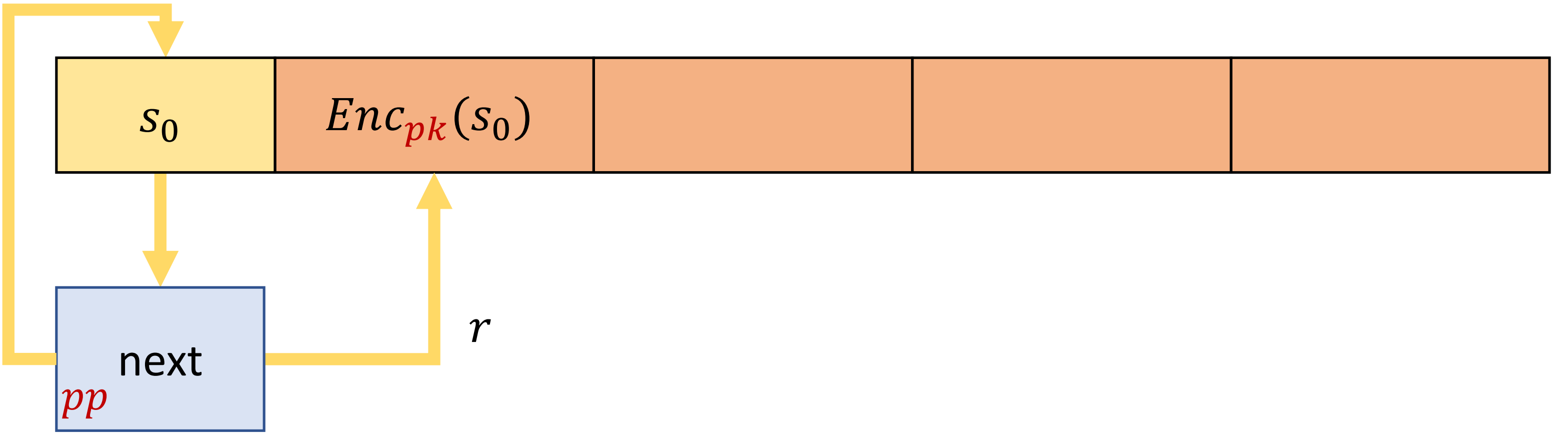
s_0

$Enc_{pk}(s_0)$

When $next$ is called

Overview of Construction

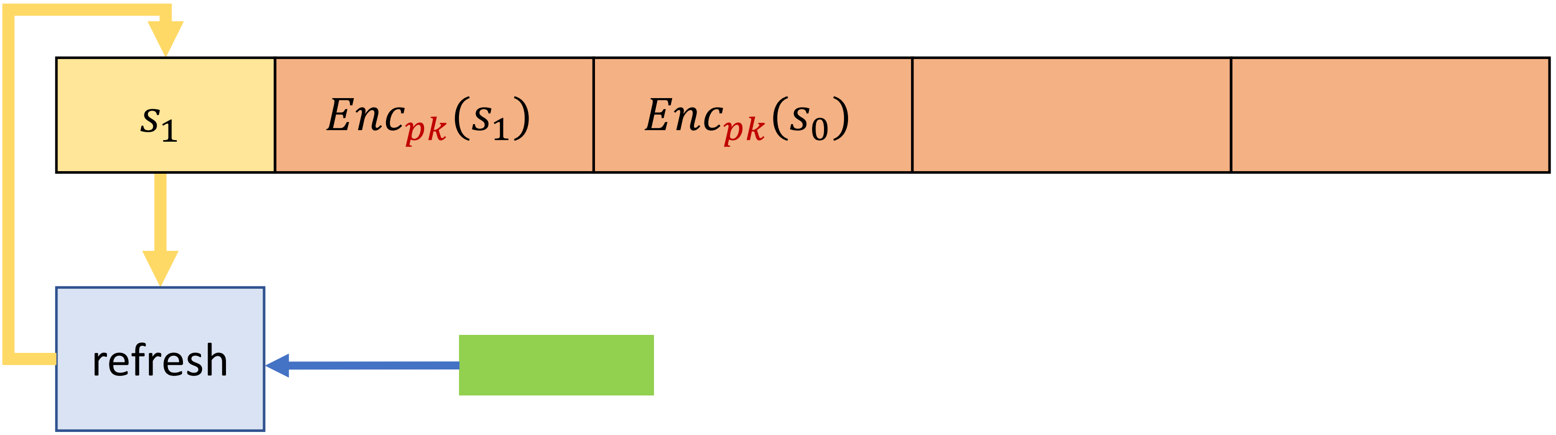
Keep snapshots of the state



When $next$ is called

Overview of Construction

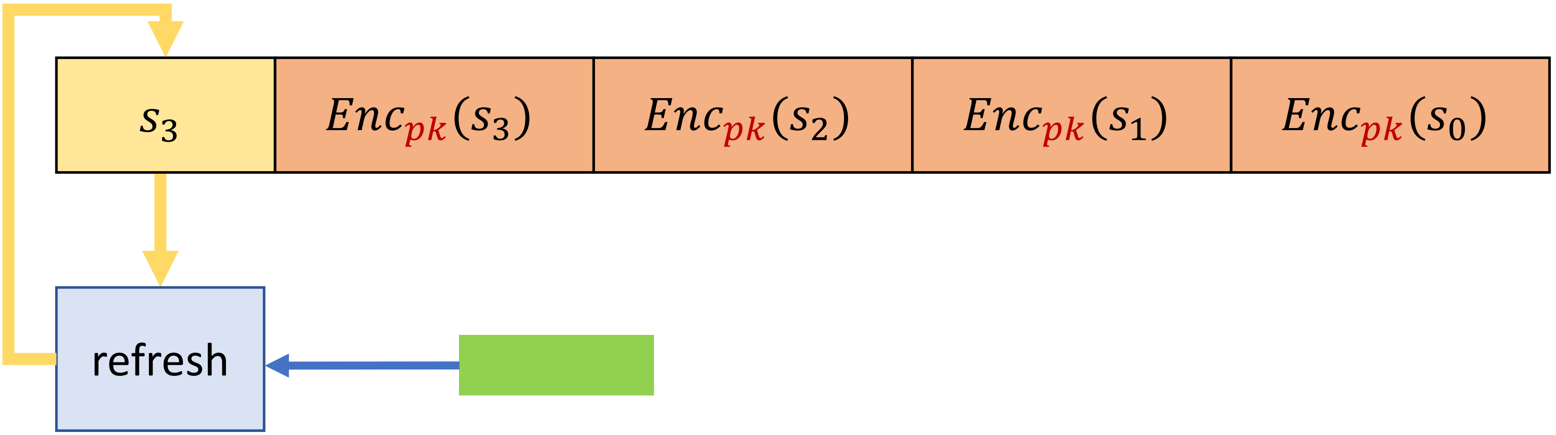
Keep snapshots of the state



When $refresh$ is called

Overview of Construction

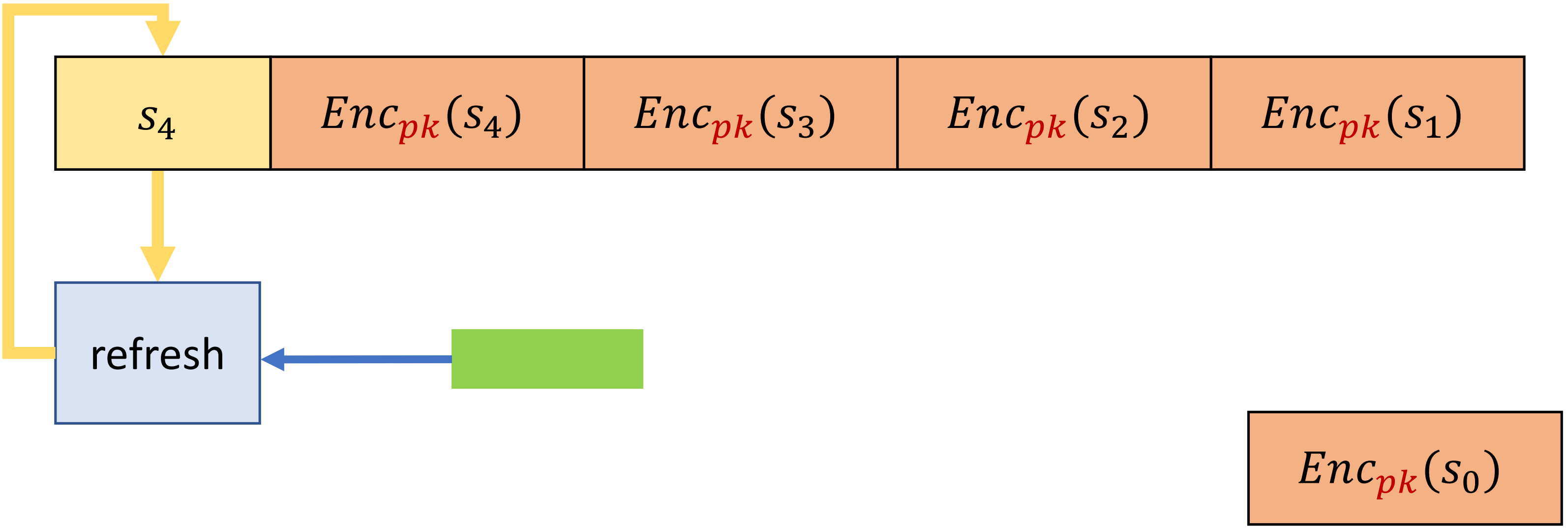
Keep snapshots of the state



When $refresh$ is called

Overview of Construction

Keep snapshots of the state

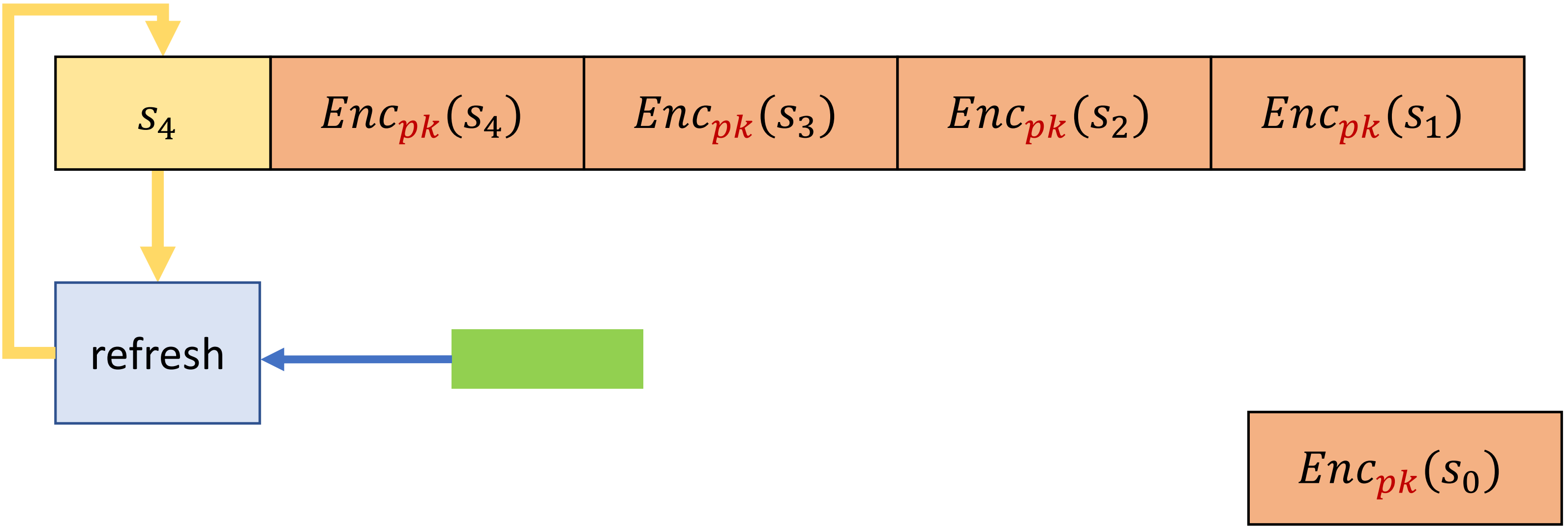


When $refresh$ is called

Overview of Construction

Keep snapshots of the state

Is it inherent that you lose information about older states?



When *refresh* is called

Impossibility

Backdooring in a strong sense cannot be achieved, while preserving robustness, without significantly enlarging the state.

Thank you. Questions?