Pseudorandom Generators Continued, Indistinguishability

CS/ECE 407

Today's objectives

Define negligible functions

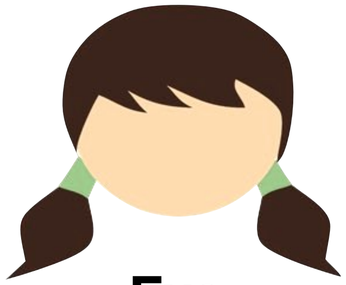
Define indistinguishability

Construct PRG with long stretch from one with short stretch

Understand proofs and hybrid worlds

Understand an attack

Alice



Eve

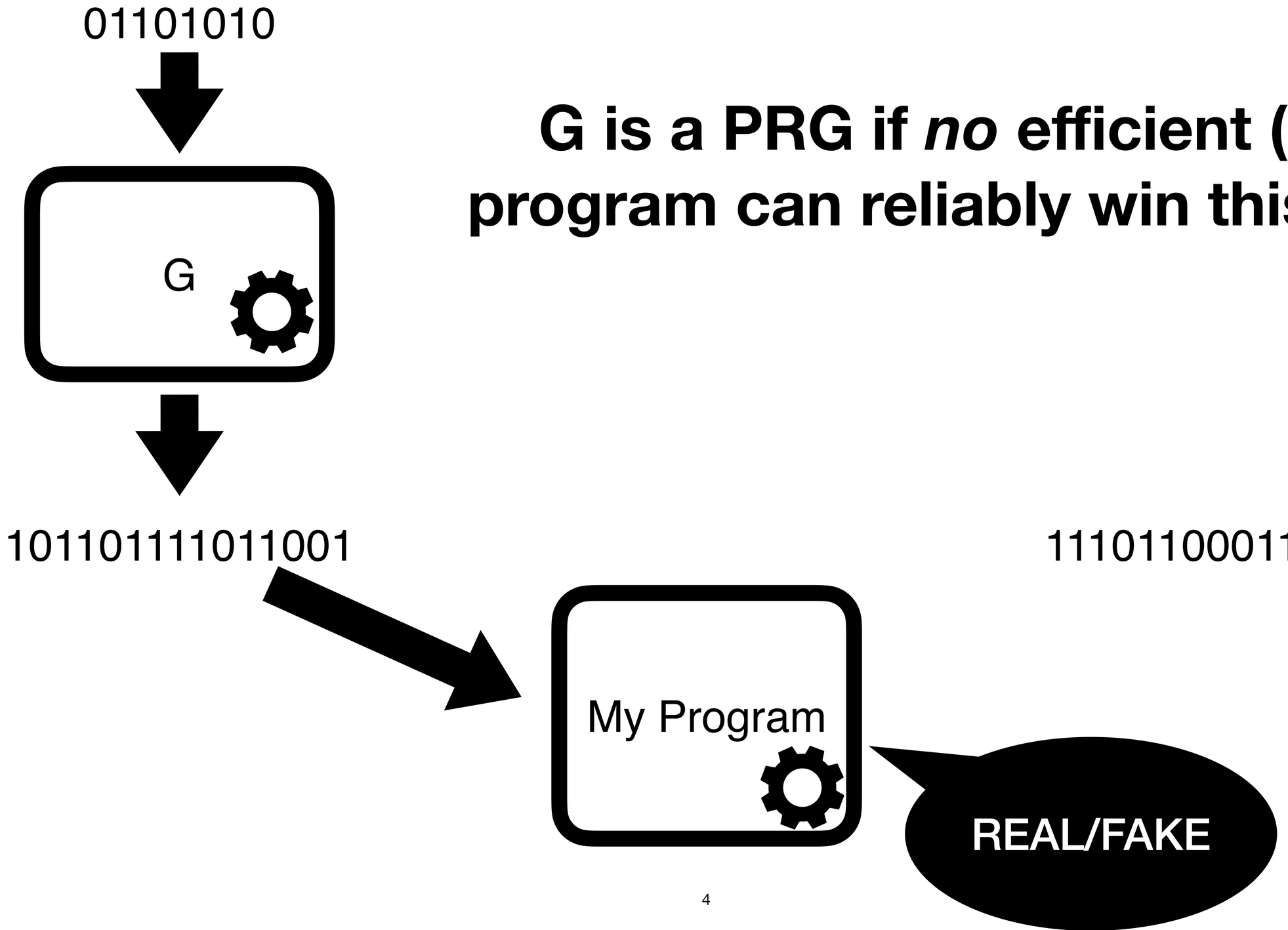
Bob

`k = 100101110`

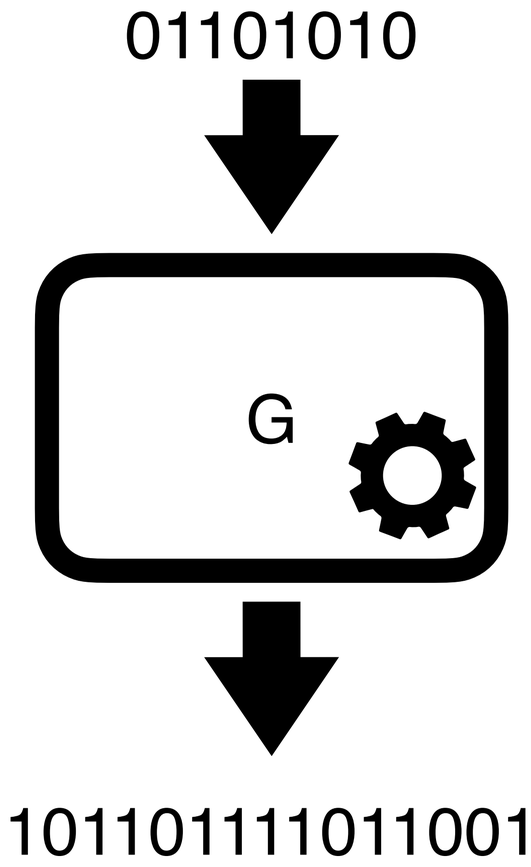
`k = 100101110`

`G(k) = 001110110100110011001`

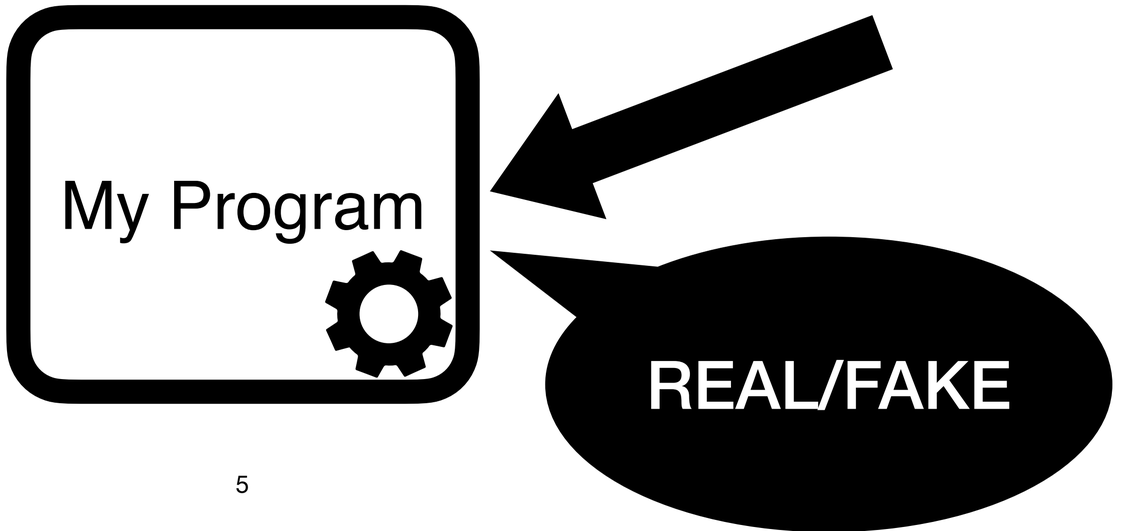
`G(k) = 001110110100110011001`



G is a PRG if *no* efficient (PPT) program can reliably win this game



G is a PRG if *no* efficient (PPT) program can reliably win this game



PRG security

```
b ←$ {0,1}
if b = 0:
   seed ←$ {0,1}^n
   y := G(seed)
else
   y ←$ {0,1}^(n+s)
b' := A(y)
```

For *any* PPT program A outputting a bit, the following quantity is **negligible** (in n):

$$\left| \Pr\left[b = b' \right] - \frac{1}{2} \right|$$

Negligible Function

A function μ is **negligible** *if for any positive polynomial p there exists an N such that for all $n > N$:*

$$\mu(n) < \frac{1}{p(n)}$$

"μ approaches zero really fast"

PRG security

```
b ←$ {0,1}
if b = 0:
   seed ←$ {0,1}^n
   y := G(seed)
else
   y ←$ {0,1}^(n+s)
b' := A(y)
```

For *any* PPT program A outputting a bit, the following quantity is **negligible** (in n):

$$\left| \Pr\left[b = b' \right] - \frac{1}{2} \right|$$

In other words, the best possible strategy is only negligibly better than simply guessing

PRG security

Game 0

$x \leftarrow\$ \{0,1\}^n$

$y := G(x)$

$b := A(y)$

Game 1

$y \leftarrow\$ \{0,1\}^{n+s}$

$b := A(y)$

For *any* PPT program A outputting a bit, the following quantity is **negligible** (in n):

$$|\Pr[\ b = 1 \mid \text{Game } 0\ ] - \Pr[\ b = 1 \mid \text{Game } 1\ ]|$$

Indistinguishability

Left

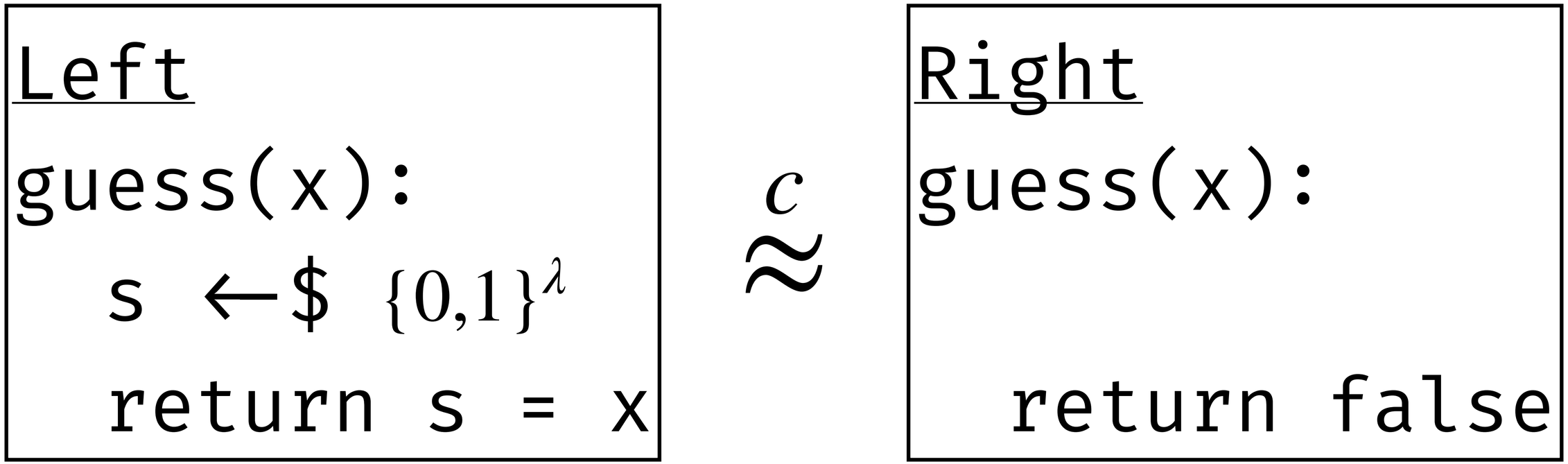
```
guess(x):
  s ←$ {0,1}^λ
  return s = x
```

Right

```
guess(x):


  return false
```

Indistinguishability



λ: the *security parameter*

Indistinguishability

```
Left
guess(x):
  s ←$ {0,1}^λ
  return s = x
```

$\overset{c}{\approx}$

```
Right
guess(x):

  return false
```

For *any* **PPT** program A outputting a bit, the following quantity is **negligible** (in λ):

$$|\Pr[\ A \diamond \text{Left} = 1\ ] - \Pr[\ A \diamond \text{Right} = 1\ ]|$$

Indistinguishability

Two programs X and Y are called **indistinguishable,** written $X \overset{c}{\approx} Y$ if for *any* **PPT** program A outputting a bit, the following quantity is **negligible** (in λ):

$$|\Pr[\ A \diamond \mathrm{X} = 1\ ] - \Pr[\ A \diamond \mathrm{Y} = 1\ ]|$$

Alternative length-doubling PRG definition

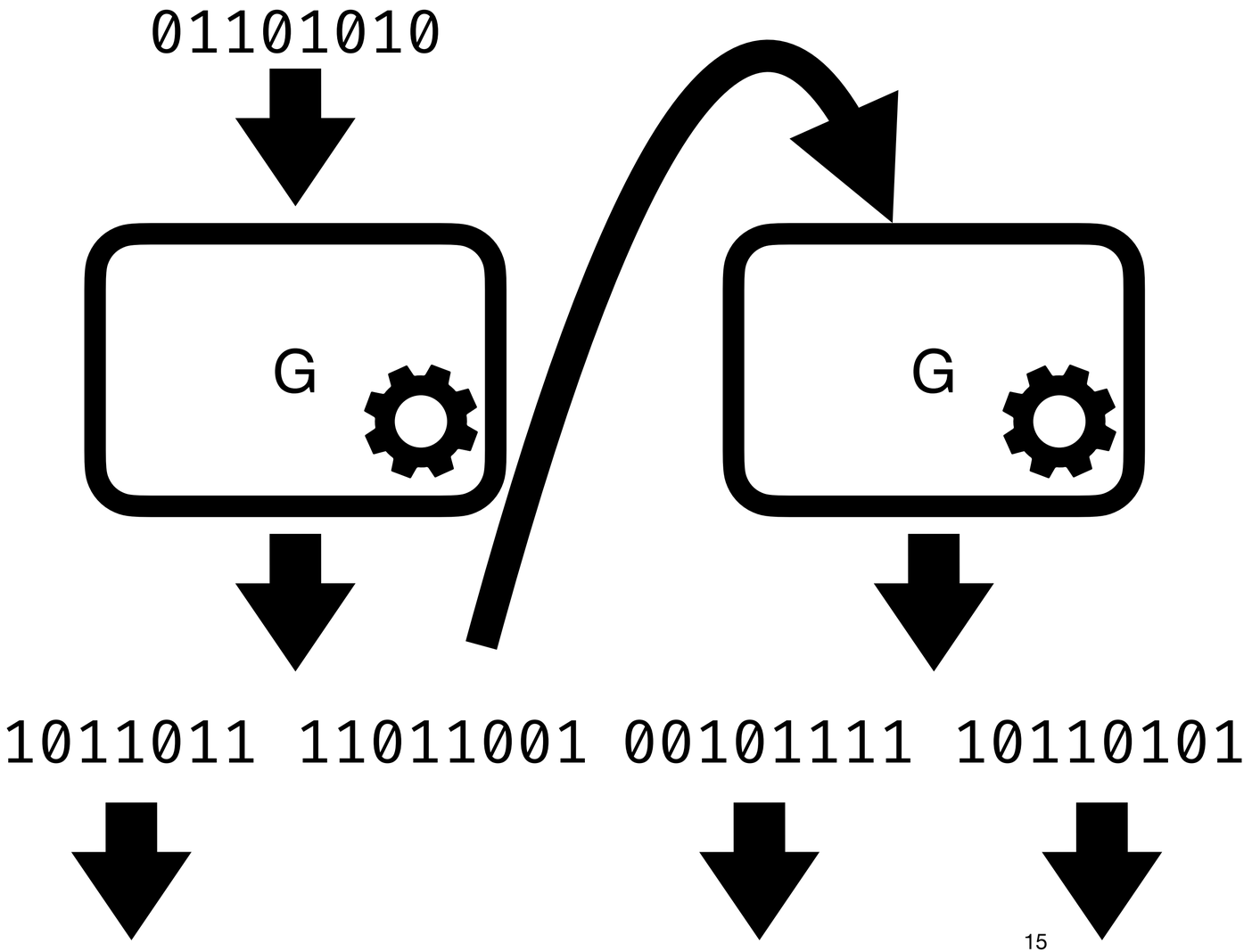
```
gen():
  x ←$ {0,1}^λ
  y ← G(x)
  return y
```

$\approx_c$

```
gen():

  y ← {0,1}^2λ
  return y
```

Stretching the output of a PRG



This is a secure PRG

Security Reduction

G is a PRG $\Longrightarrow$ G’ is a PRG

G is a not PRG $\Longleftarrow$ G’ is not a PRG

```
gen():
  x ←$ {0,1}^λ
  y ← G(x)
  return y
```

$\overset{c}{\approx}$

```
gen():

  y ← {0,1}^2λ
  return y
```

```
gen-long():
  x ←$ {0,1}^λ
  y ← G'(x)
  return y
```

$\overset{c}{\approx}$

```
gen-long():

  y ← {0,1}^3λ
  return y
```

Stretching the output of a PRG

```
gen-long():
   s_0 ←$ {0,1}^λ
   x || y || z ← G'(s_0)
   return x || y || z
```

Stretching the output of a PRG

```
gen-long():
   s_0 ←$ {0,1}^λ
   x || s_1 ← G(s_0)
   y || z ← G(s_1)
   return x || y || z
```

Stretching the output of a PRG

```
gen-long():
  s_0 ←$ {0,1}^λ
  x || s_1 ← G(s_0)
  y || z ← G(s_1)
  return x || y || z
```

=

```
gen-long():
  x || s_1 ← gen()
  y || z ← G(s_1)
  return x || y || z
```

◇

```
gen():
  s ←$ {0,1}^λ
  r ← G(s)
  return r
```

Stretching the output of a PRG

```
gen-long():
  x || s_1 ← gen()
  y || z ← G(s_1)
  return x || y || z
```

◇

```
gen():
  s ←$ {0,1}^λ
  r ← G(s)
  return r
```

Stretching the output of a PRG

```
gen-long():
  x || s_1 ← gen()
  y || z ← G(s_1)
  return x || y || z
```

$\diamond$

```
gen():
  s ←$ {0,1}^λ
  r ← G(s)
  return r
```

$\overset{c}{\approx}$

By PRG indistinguishability

```
gen-long():
  x || s_1 ← gen()
  y || z ← G(s_1)
  return x || y || z
```

$\diamond$

```
gen():

  r ←$ {0,1}^(2·λ)
  return r
```

Stretching the output of a PRG

```
gen-long():
  x || s_1 ← gen()
  y || z ← G(s_1)
  return x || y || z
```

◇

```
gen():
  s ←$ {0,1}^λ
  r ← G(s)
  return r
```

$\overset{c}{\approx}$

“Hybrid World”

```
gen-long():
  x || s_1 ← gen()
  y || z ← G(s_1)
  return x || y || z
```

◇

```
gen():

  r ←$ {0,1}^(2·λ)
  return r
```

Stretching the output of a PRG

```
gen-long():
  x || s_1 ← gen()
  y || z ← G(s_1)
  return x || y || z
```

◇

```
gen():

  r ←$ {0,1}^{2·λ}
  return r
```

Stretching the output of a PRG

```
gen-long():
  x || s1 ← gen()
  y || z ← G(s1)
  return x || y || z
```

◇

```
gen():

  r ←$ {0,1}^(2·λ)
  return r
```

=

```
gen-long():
  x || s1 ←$ {0,1}^(2·λ)
  y || z ← G(s1)
  return x || y || z
```

Stretching the output of a PRG

```
gen-long():
  x || s_1 ← gen()
  y || z ← G(s_1)
  return x || y || z
```

◇

```
gen():
  r ←$ {0,1}^{2·λ}
  return r
```

=

```
gen-long():
  x || s_1 ←$ {0,1}^{2·λ}
  y || z ← G(s_1)
  return x || y || z
```

=

```
gen-long():
  x ←$ {0,1}^λ
  s_1 ←$ {0,1}^λ
  y || z ← G(s_1)
  return x || y || z
```

Stretching the output of a PRG

```
gen-long():
   x ←$ {0,1}^λ
   s_1 ←$ {0,1}^λ
   y || z ← G(s_1)
   return x || y || z
```

Stretching the output of a PRG

```
gen-long():
  x ←$ {0,1}^λ
  s_1 ←$ {0,1}^λ
  y || z ← G(s_1)
  return x || y || z
```

=

```
gen-long():
  x ←$ {0,1}^λ
  y || z ← gen()
  return x || y || z
```

◇

```
gen():
  s ←$ {0,1}^λ
  r ← G(s)
  return r
```

Stretching the output of a PRG

```
gen-long():
  x ←$ {0,1}^λ
  y || z ← gen()
  return x || y || z
```

◇

```
gen():
  s ←$ {0,1}^λ
  r ← G(s)
  return r
```

Stretching the output of a PRG

```
gen-long():
  x ←$ {0,1}^λ
  y || z ← gen()
  return x || y || z
```

◇

```
gen():
  s ←$ {0,1}^λ
  r ← G(s)
  return r
```

$\overset{c}{\approx}$

By PRG indistinguishability

```
gen-long():
  x ←$ {0,1}^λ
  y || z ← gen()
  return x || y || z
```

◇

```
gen():

  r ←$ {0,1}^{2·λ}
  return r
```

Stretching the output of a PRG

```
gen-long():
  x ←$ {0,1}^λ
  y || z ← gen()
  return x || y || z
```

◇

```
gen():

  r ←$ {0,1}^(2·λ)
  return r
```

Stretching the output of a PRG

```
gen-long():
  x ←$ {0,1}^λ
  y || z ← gen()
  return x || y || z
```

◇

```
gen():

  r ←$ {0,1}^(2·λ)
  return r
```

=

```
gen-long():
  x || y || z ←$ {0,1}^(3λ)
  return x || y || z
```

Stretching the output of a PRG

```
gen-long():
  x ←$ {0,1}^λ
  y || z ← gen()
  return x || y || z
```

◇

```
gen():

  r ←$ {0,1}^(2·λ)
  return r
```

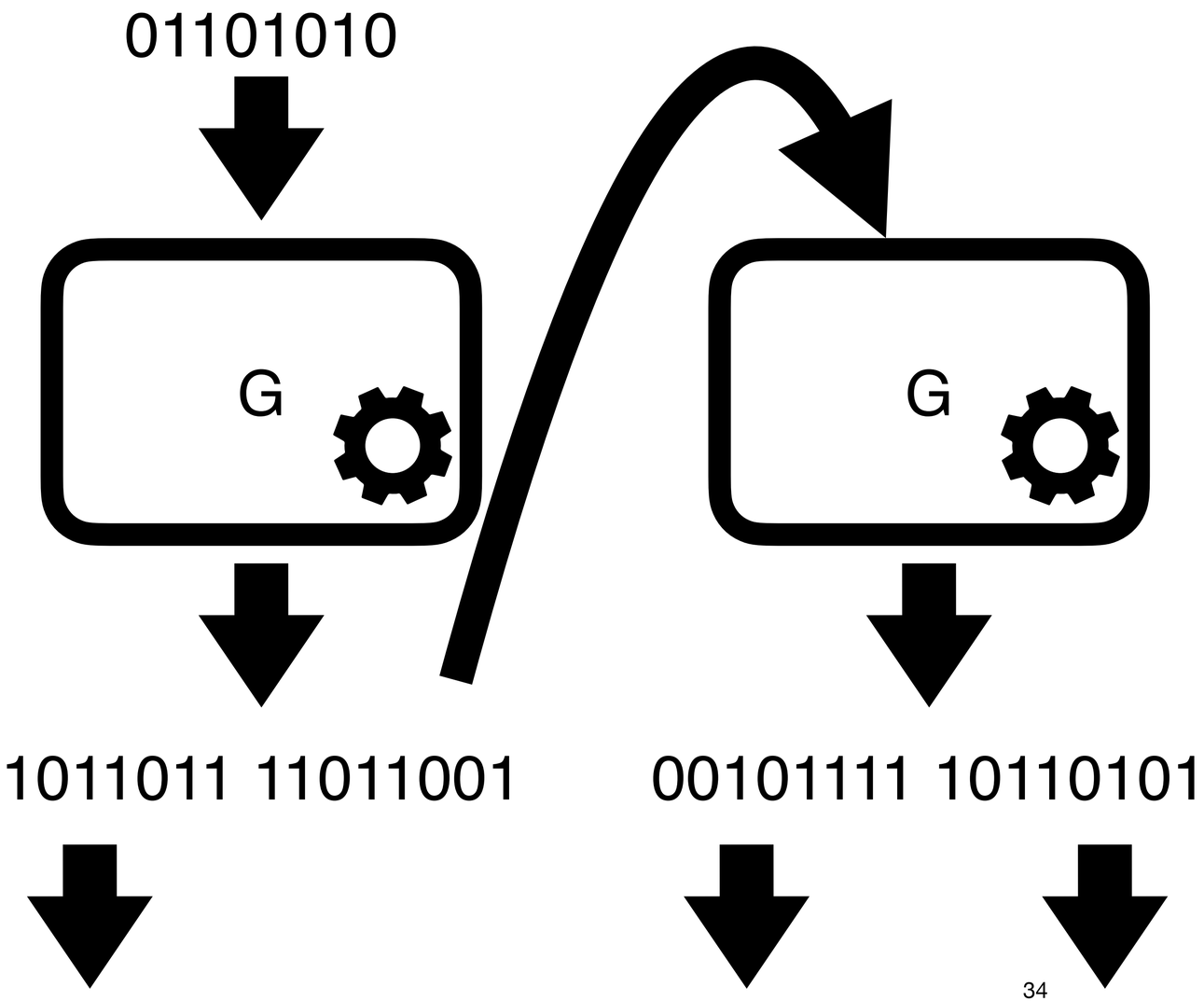
=

```
gen-long():
  x || y || z ←$ {0,1}^3λ
  return x || y || z
```

=

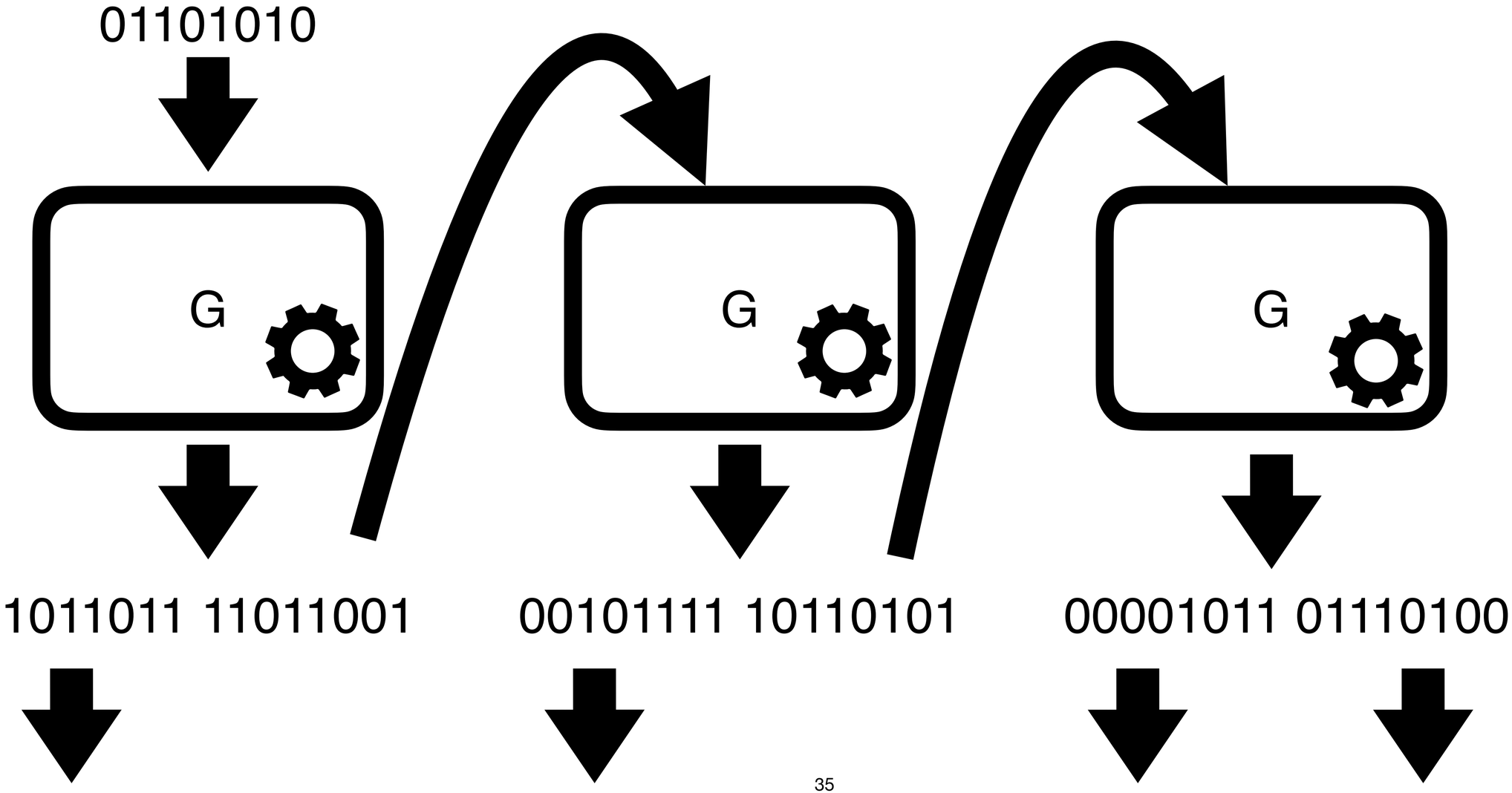
```
gen-long():
  r ←$ {0,1}^3λ
  return r
```

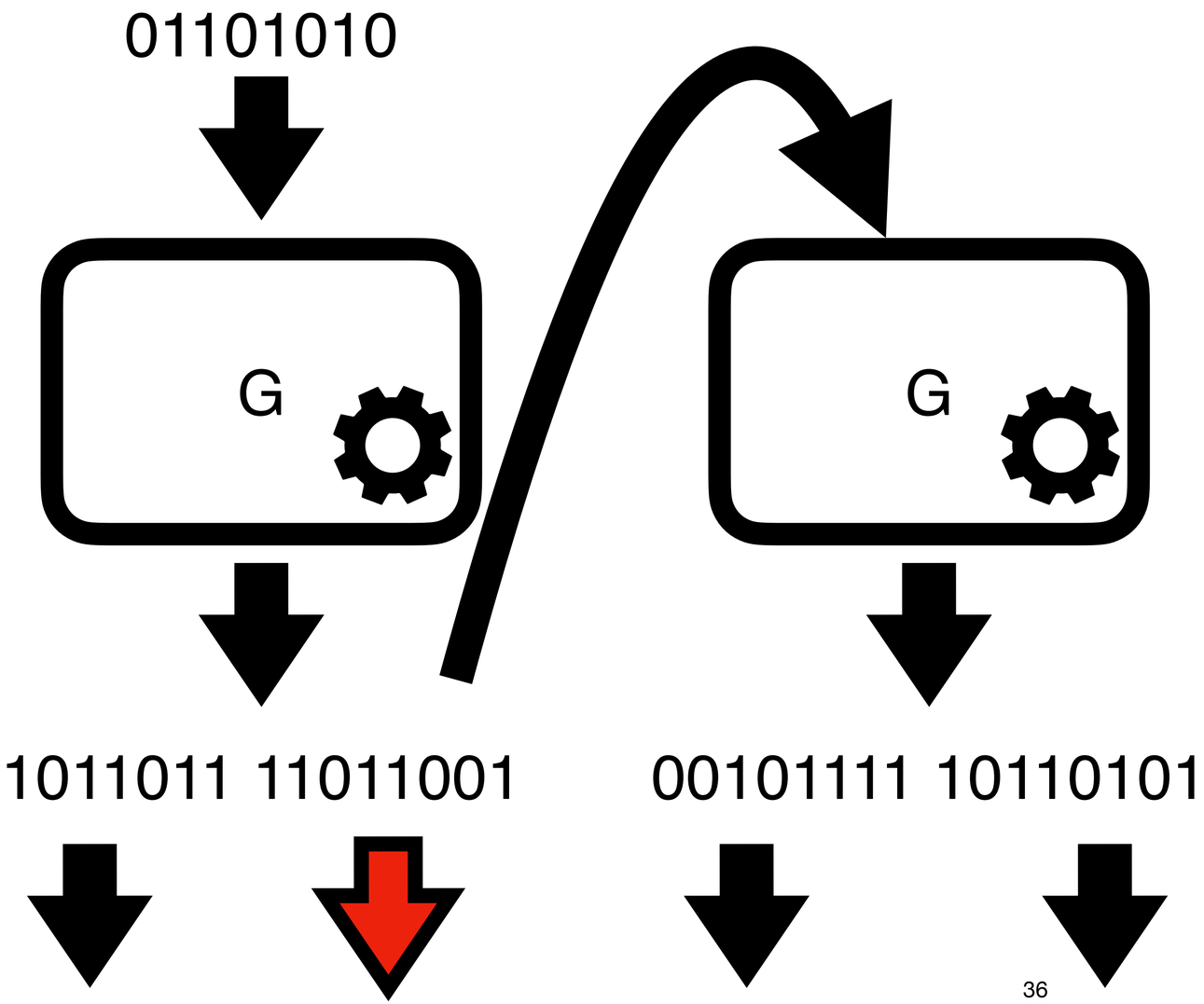
Stretching the output of a PRG



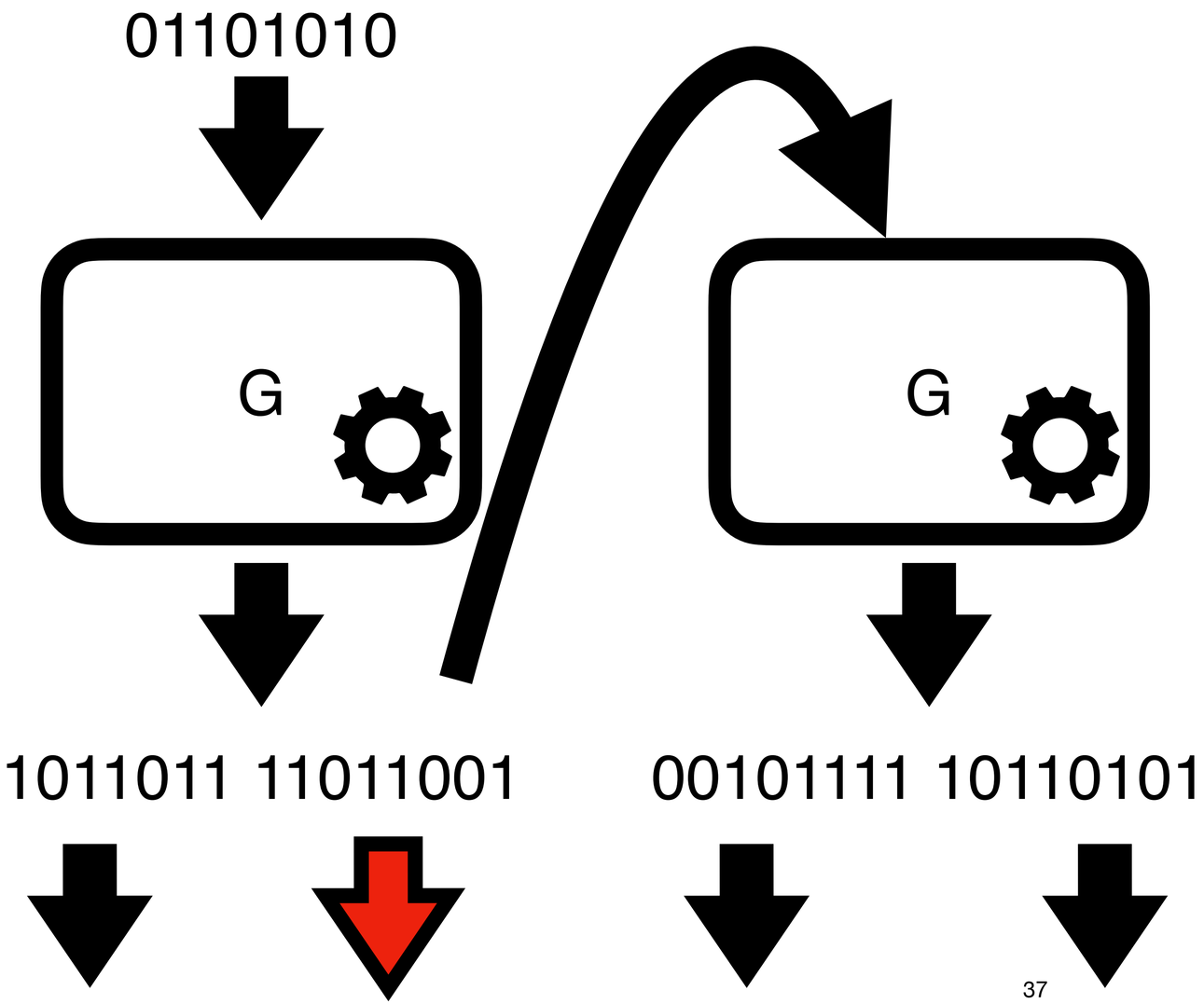
This is a secure PRG

Repeatable any polynomial number of times





Is this secure?




```
gen-long():
  s ←$ {0,1}^λ
  w || x ← G(s)
  y || z ← G(x)
  return w || x || y || z
```

Today's objectives

Understand alternative definitions of PRG security

Define security parameter

Define indistinguishability

Construct PRG with long stretch from one with short stretch

Understand an attack