Pseudorandom Generators

CS/ECE 407

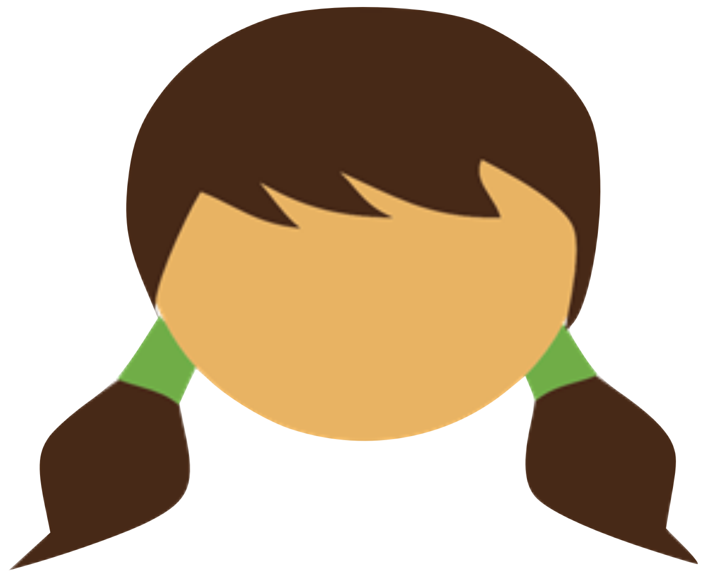
Today's objectives

Describe pseudorandomness/pseudorandom generators

Define negligible functions

Introduce indistinguishability

ct

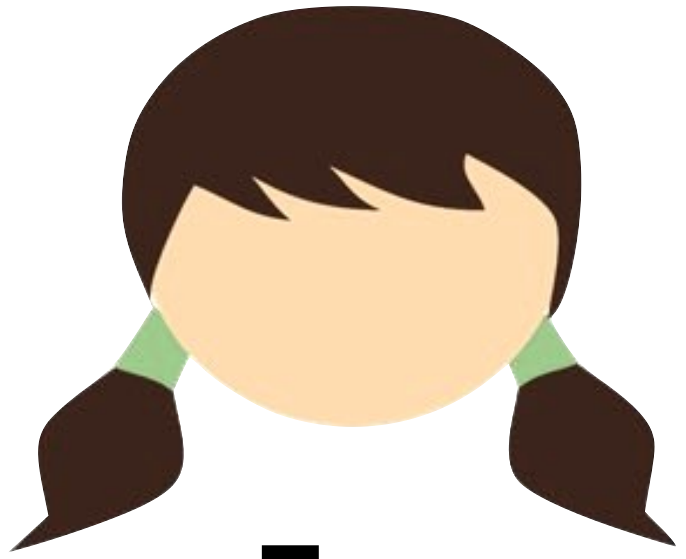


Alice

$m \in \{0,1\}$

$k \leftarrow_{\$} \{0,1\}$

$ct \leftarrow m \oplus k$

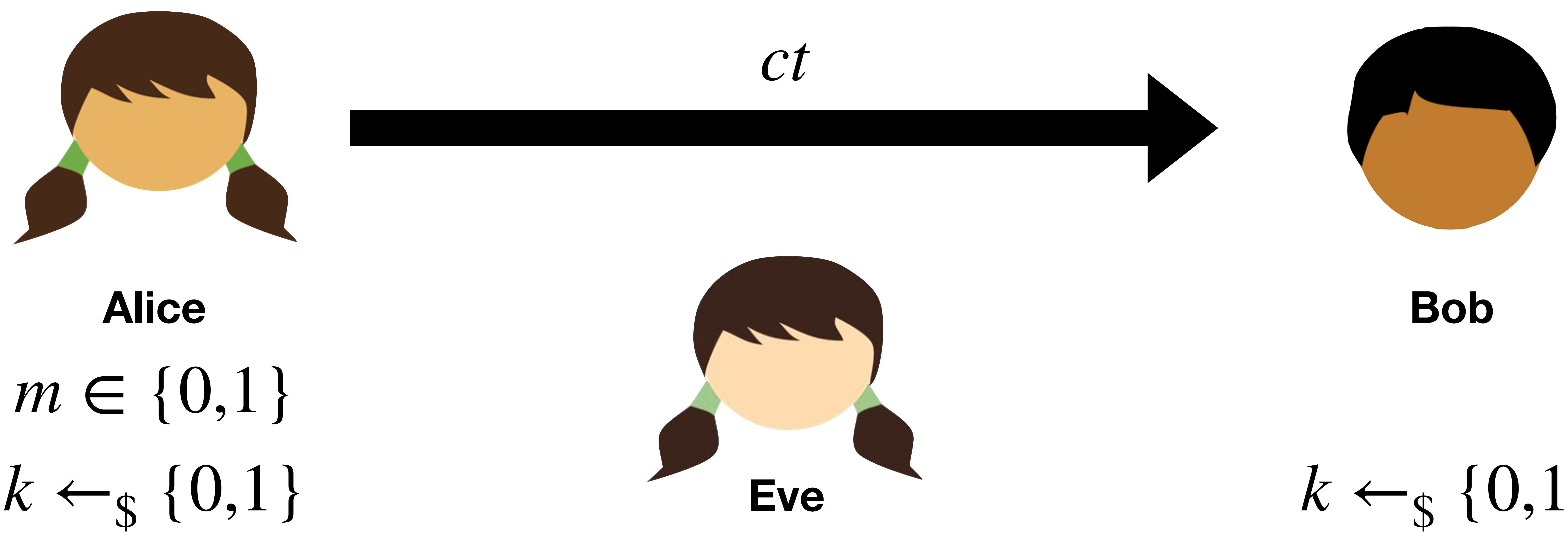


Eve

Bob

$k \leftarrow_{\$} \{0,1\}$

$m' \leftarrow ct \oplus k$



Alice

$m \in \{0,1\}$

$k \leftarrow_{\$} \{0,1\}$

$ct \leftarrow m \oplus k$

Bob

$k \leftarrow_{\$} \{0,1\}$

$m' \leftarrow ct \oplus k$

Question: *what if Alice wants to send more than one bit?*

Perfect Secrecy:

A cipher (Enc, Dec) is **perfectly secret** if for every message $m \in \mathrm{M}$:

$$\left\{ c \;\middle|\; \begin{array}{l} k \leftarrow_{\$} K \\ c = Enc(k, m) \end{array} \right\} \equiv \left\{ c \;\middle|\; c \leftarrow_{\$} C \right\}$$

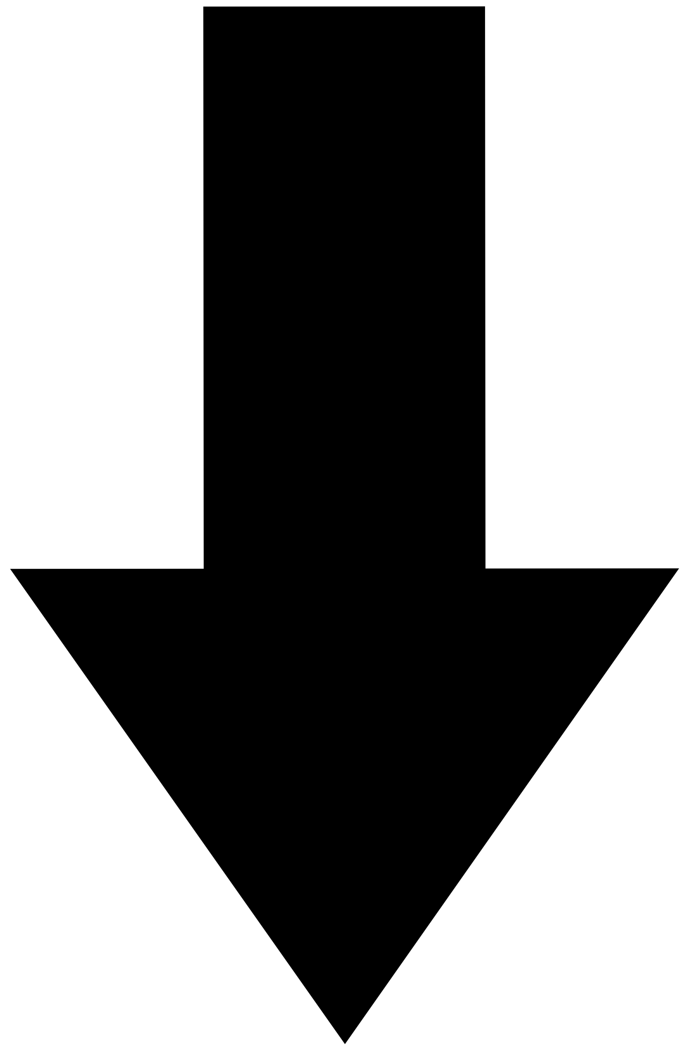
Theorem [Shannon 1949]: *Any cipher achieving perfect secrecy requires that* $|\mathrm{K}| \geq |\mathrm{M}|$*.*

“If we want to encrypt more stuff, we need more randomness”

Theorem [Shannon 1949]: *Any cipher achieving perfect secrecy requires that* $|\mathrm{K}| \geq |\mathrm{M}|$.

“If we want to encrypt more stuff, we need more randomness”

011010100

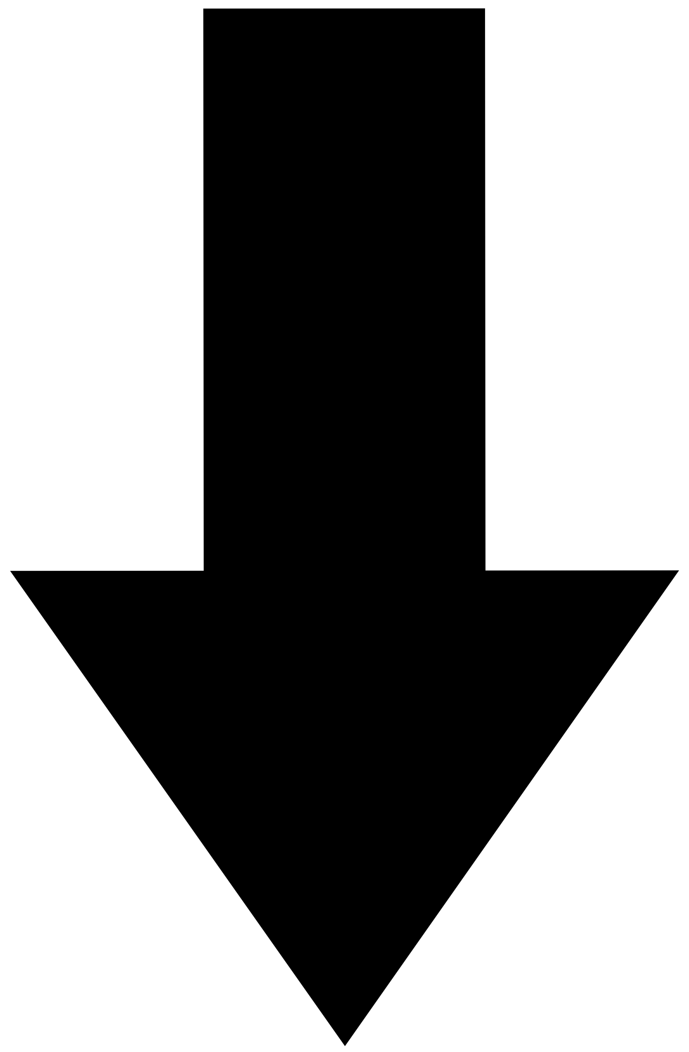


101101111011001

Q: Can we turn a short random string into a long random string?

“If we want to encrypt more stuff, we need more randomness”

011010100

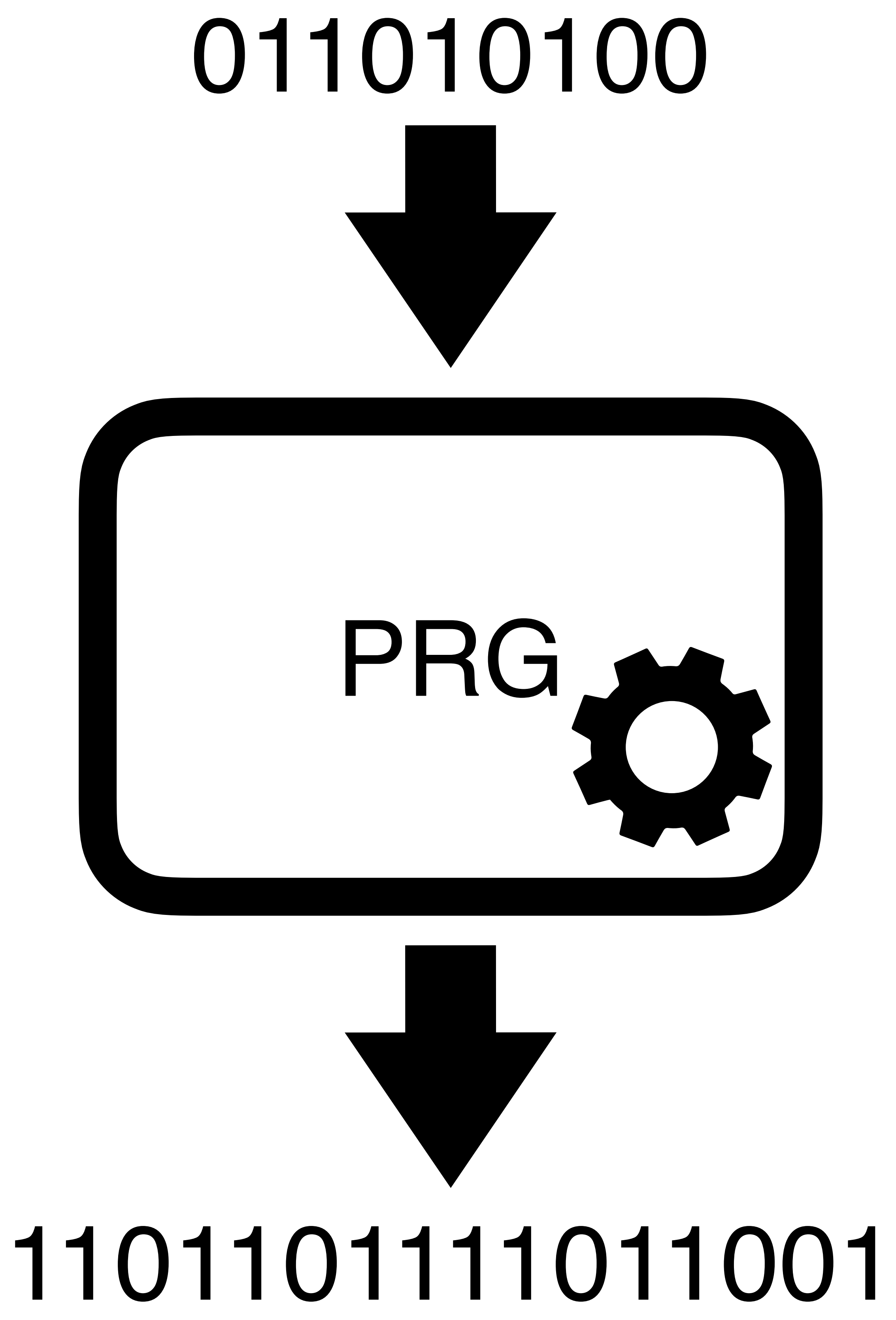


101101111011001

Q: Can we turn a short random string into a long random string?

A: No, this is impossible

"If we want to encrypt more stuff, we need more randomness"



Q: Can we turn a short random string into a long random string?

A: No, this is impossible

Q: Can we turn a short random string into a long string that <u>looks</u> random?

A: Yes[†]! Use a pseudorandom generator!

† Or, at least, we believe this to be possible

Pseudorandom Generator (PRG)

A PRG is a function $G : \{0,1\}^n \rightarrow \{0,1\}^{n+s}$

Pseudorandom Generator (PRG)

A PRG is a function $G : \{0,1\}^n \rightarrow \{0,1\}^{n+s}$

Security?

Pseudorandom Generator (PRG)

A PRG is a function $G : \{0,1\}^n \rightarrow \{0,1\}^{n+s}$

Security?

Informal: *"no program can tell the difference between the output of* G *and truly random strings"*

Hardness as a basis for cryptography

Security?

Informal: *"no program can tell the difference between the output of* G *and truly random strings"*

Modern Cryptography

State assumptions

Define security

Design system

Prove: if assumption holds, system meets definition

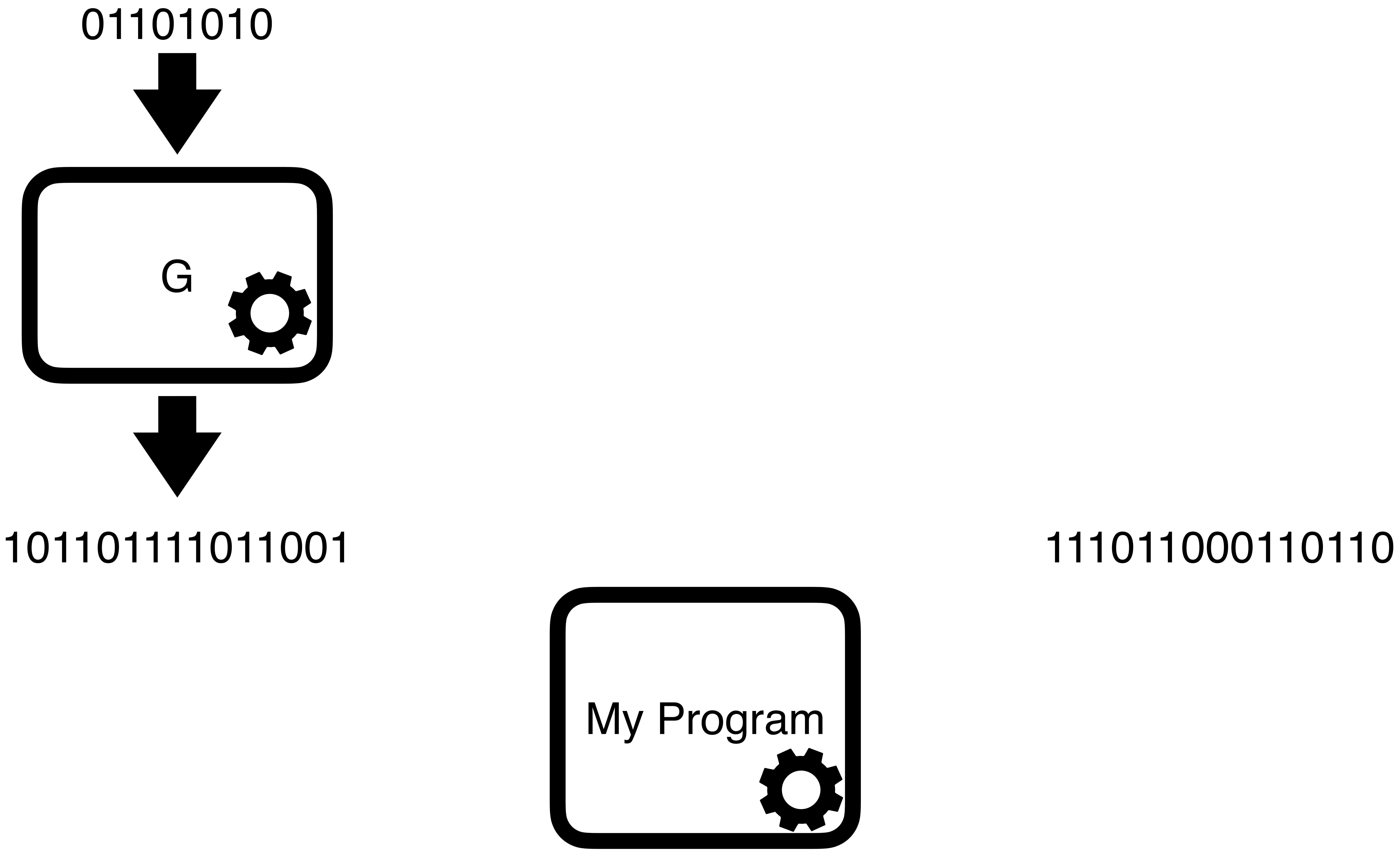
Modern Cryptography

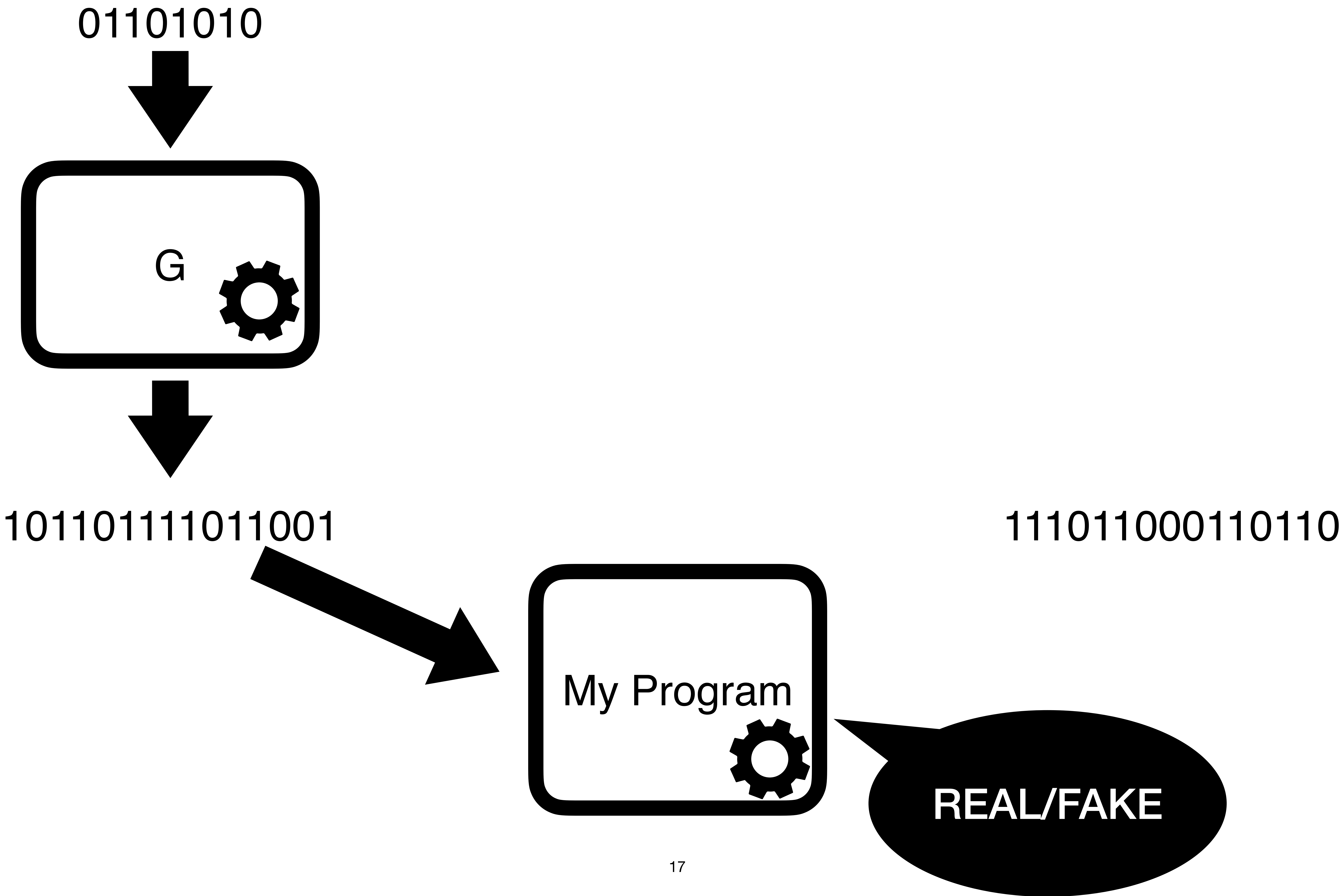
State assumptions **PRGs exist**

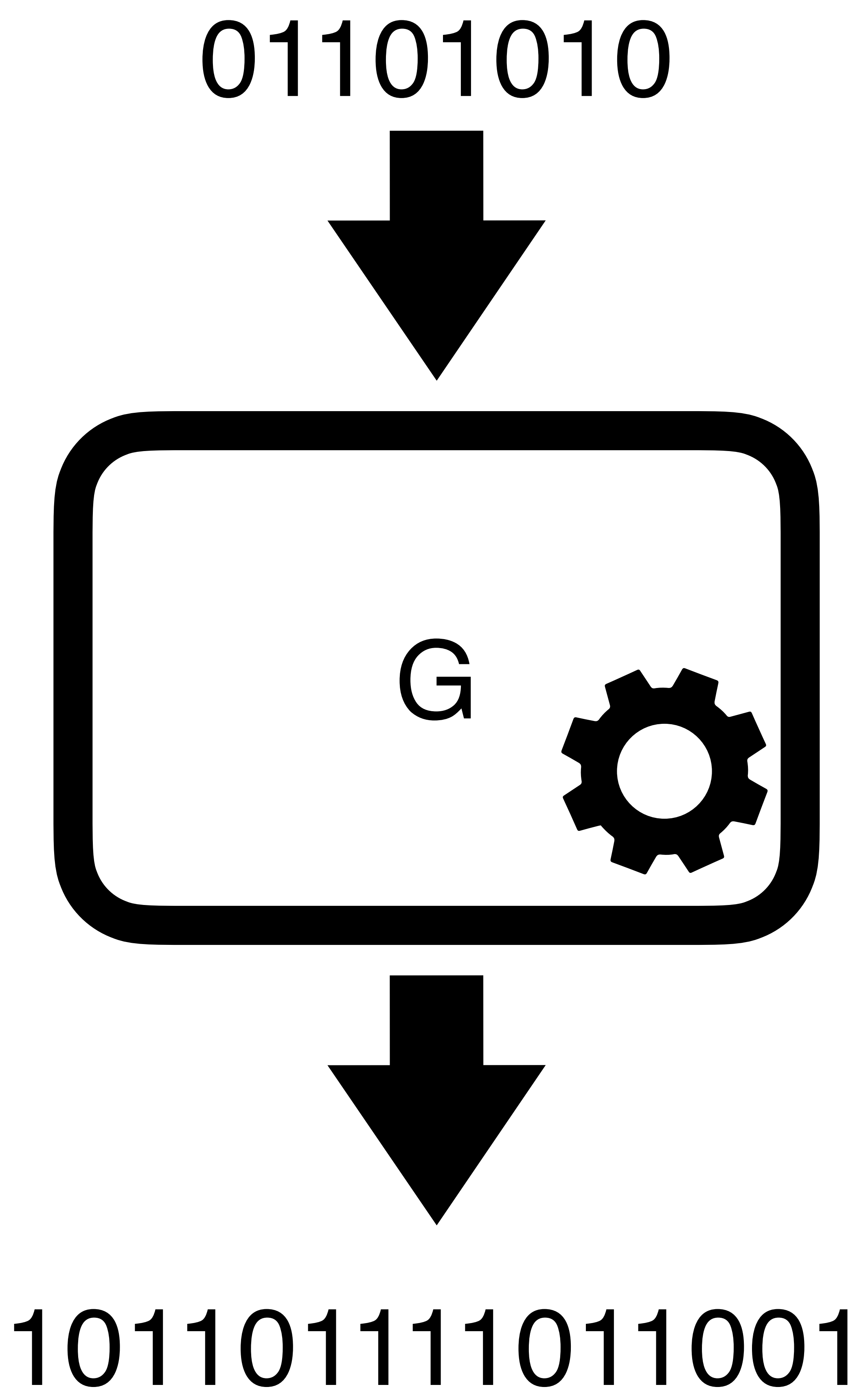
Define security

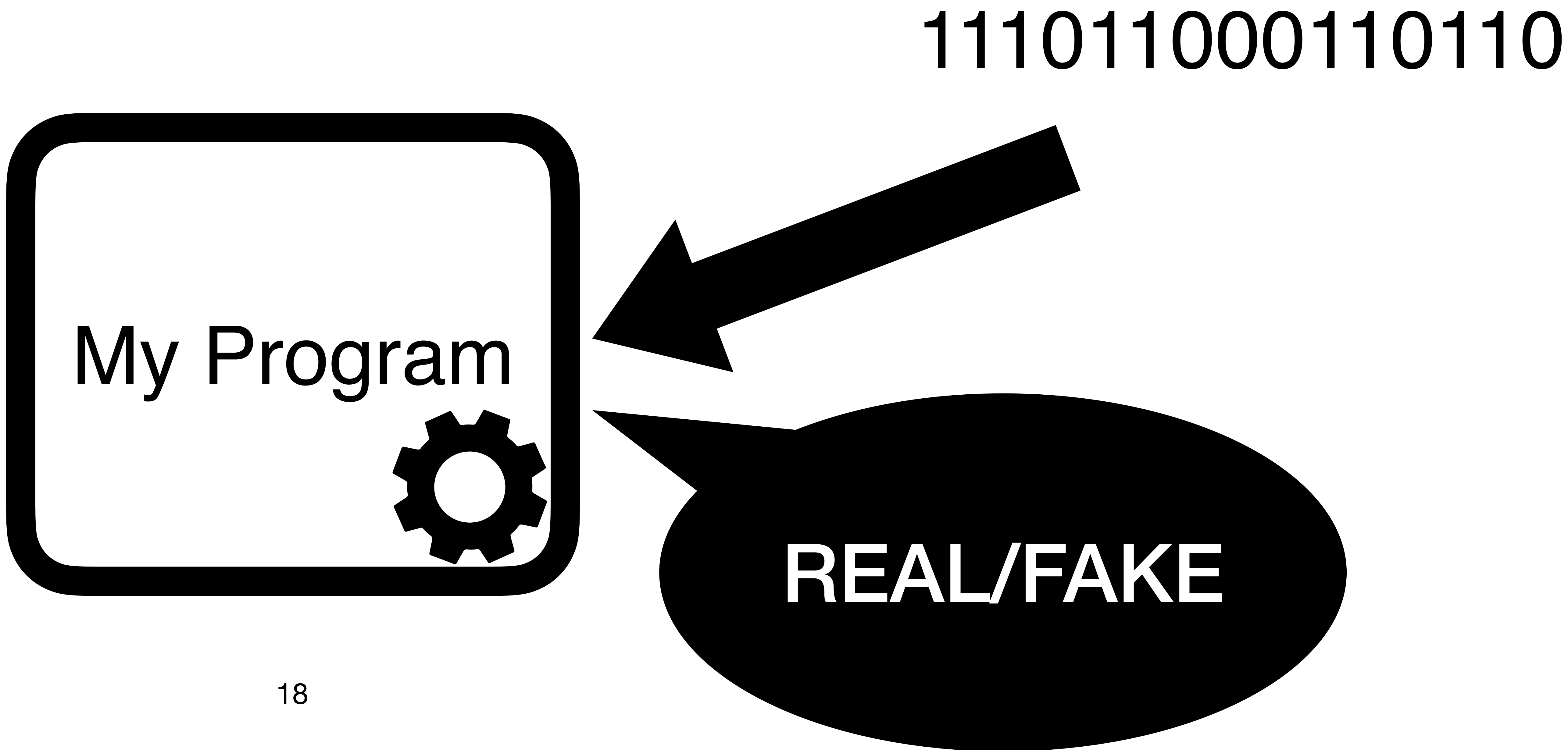
Design system

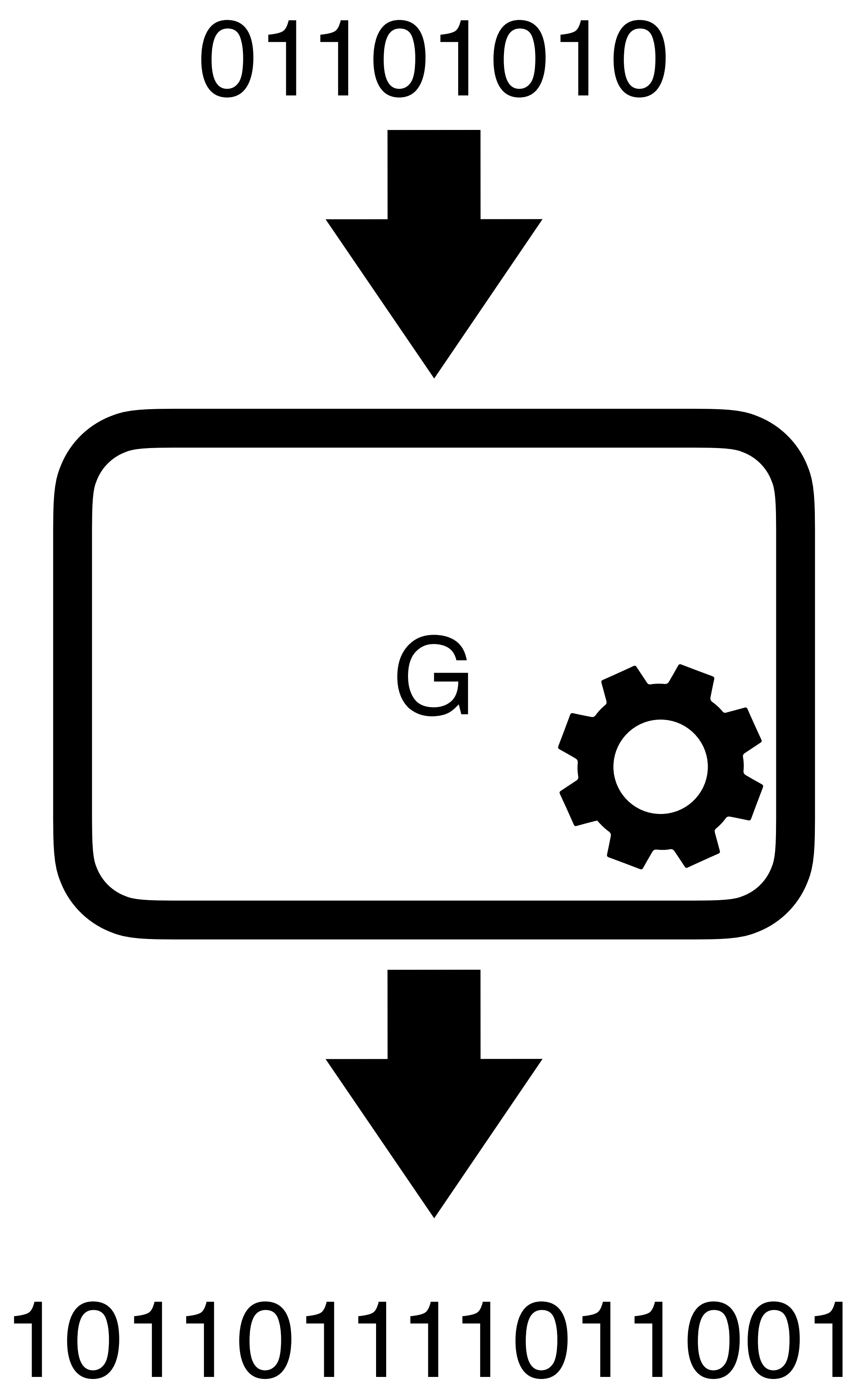
Prove: if assumption holds, system meets definition


01101010
G
101101111011001
111011000110110
My Program

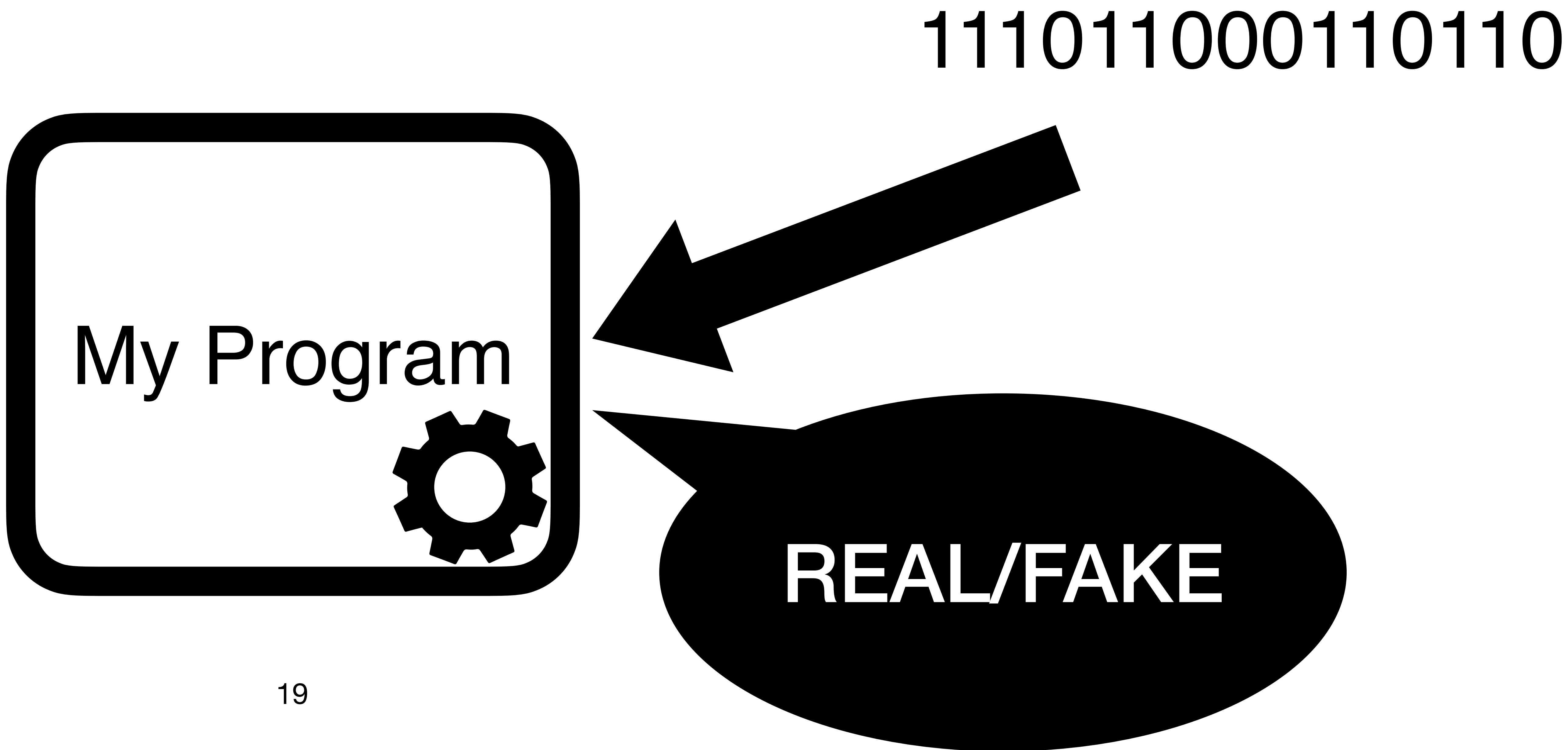

01101010
G
101101111011001
111011000110110
My Program
REAL/FAKE

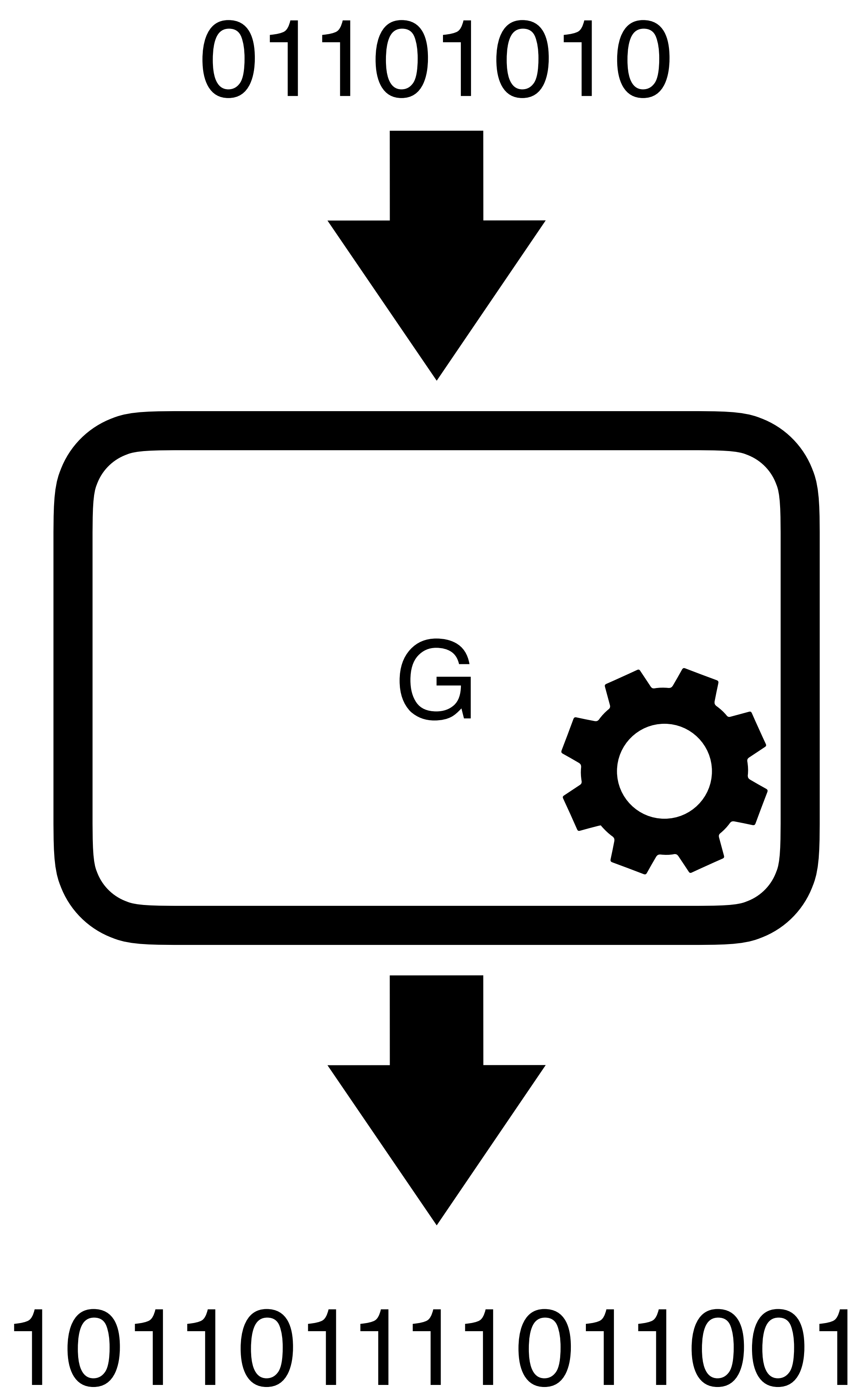

01101010
G
101101111011001


111011000110110
My Program
REAL/FAKE

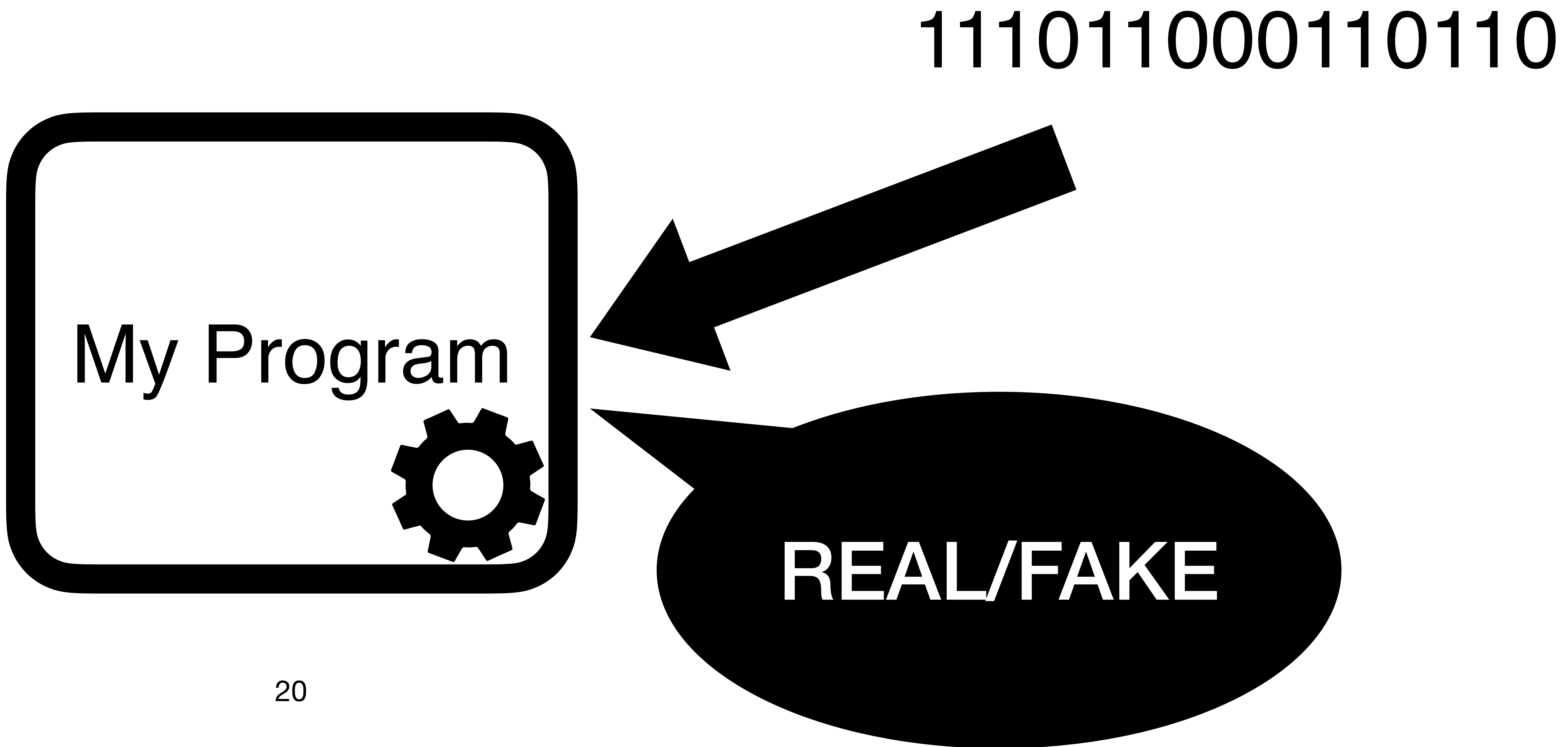


G is a PRG if *no* program can reliably win this game

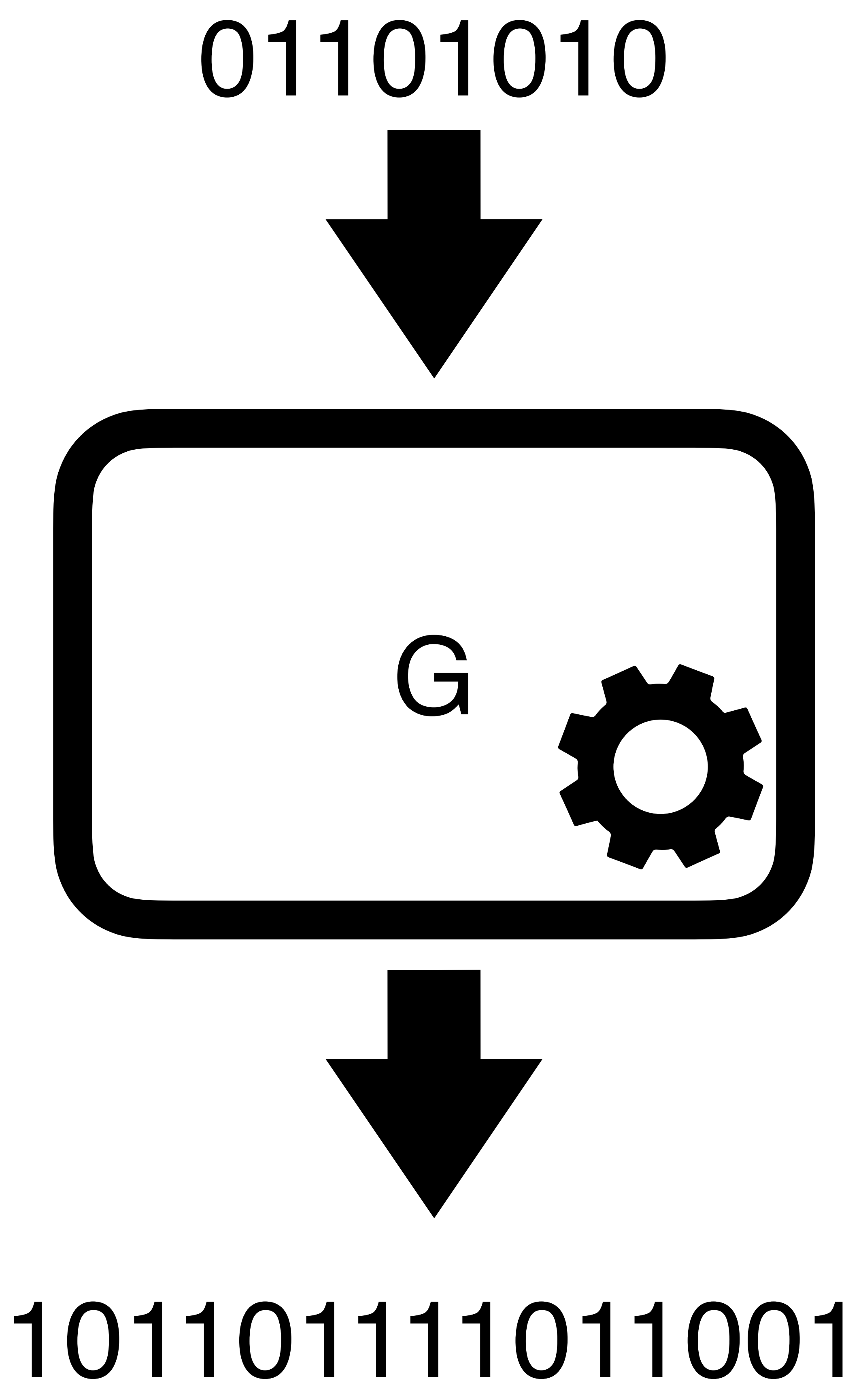




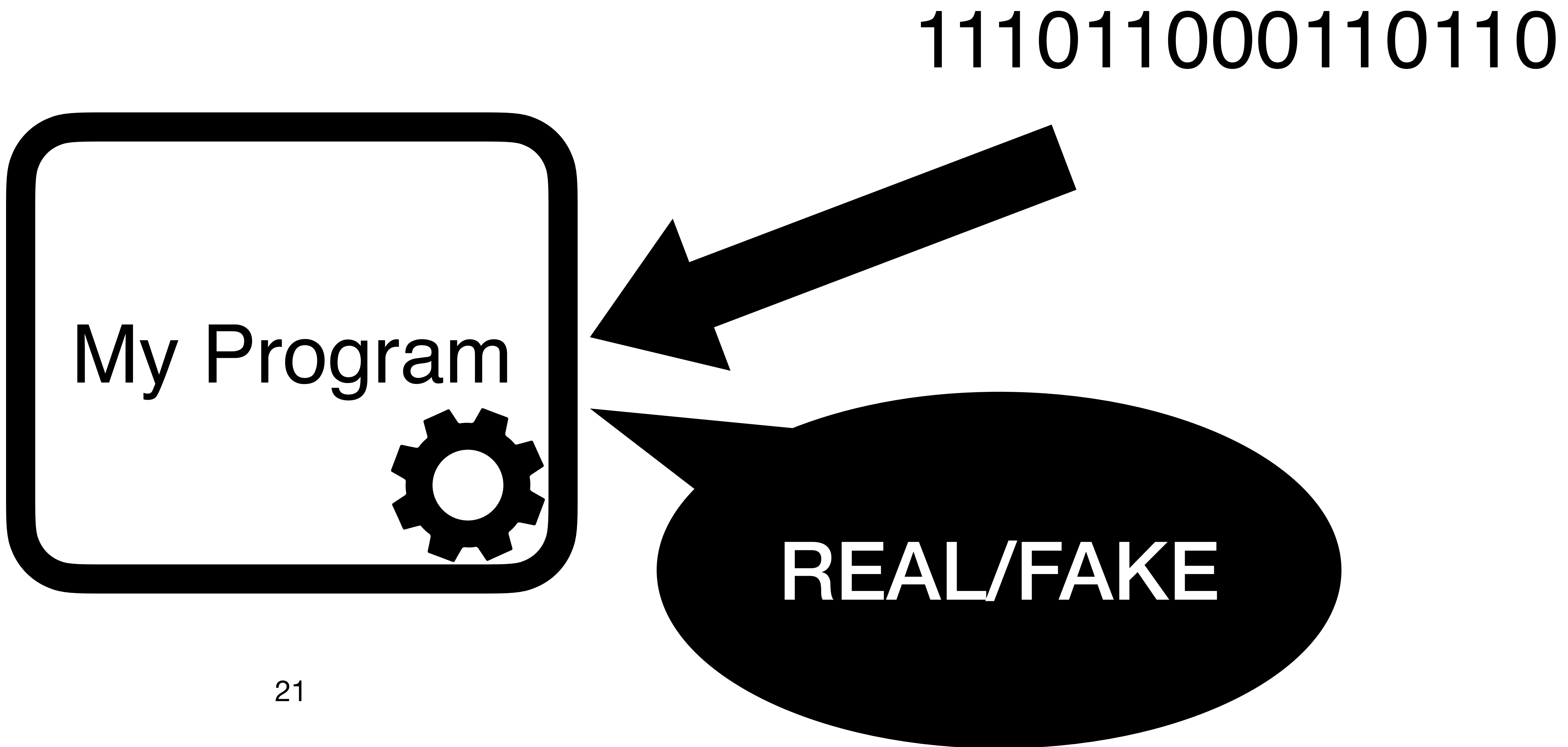
G is a PRG if *no* program can reliably win this game



We believe that PRGs exist

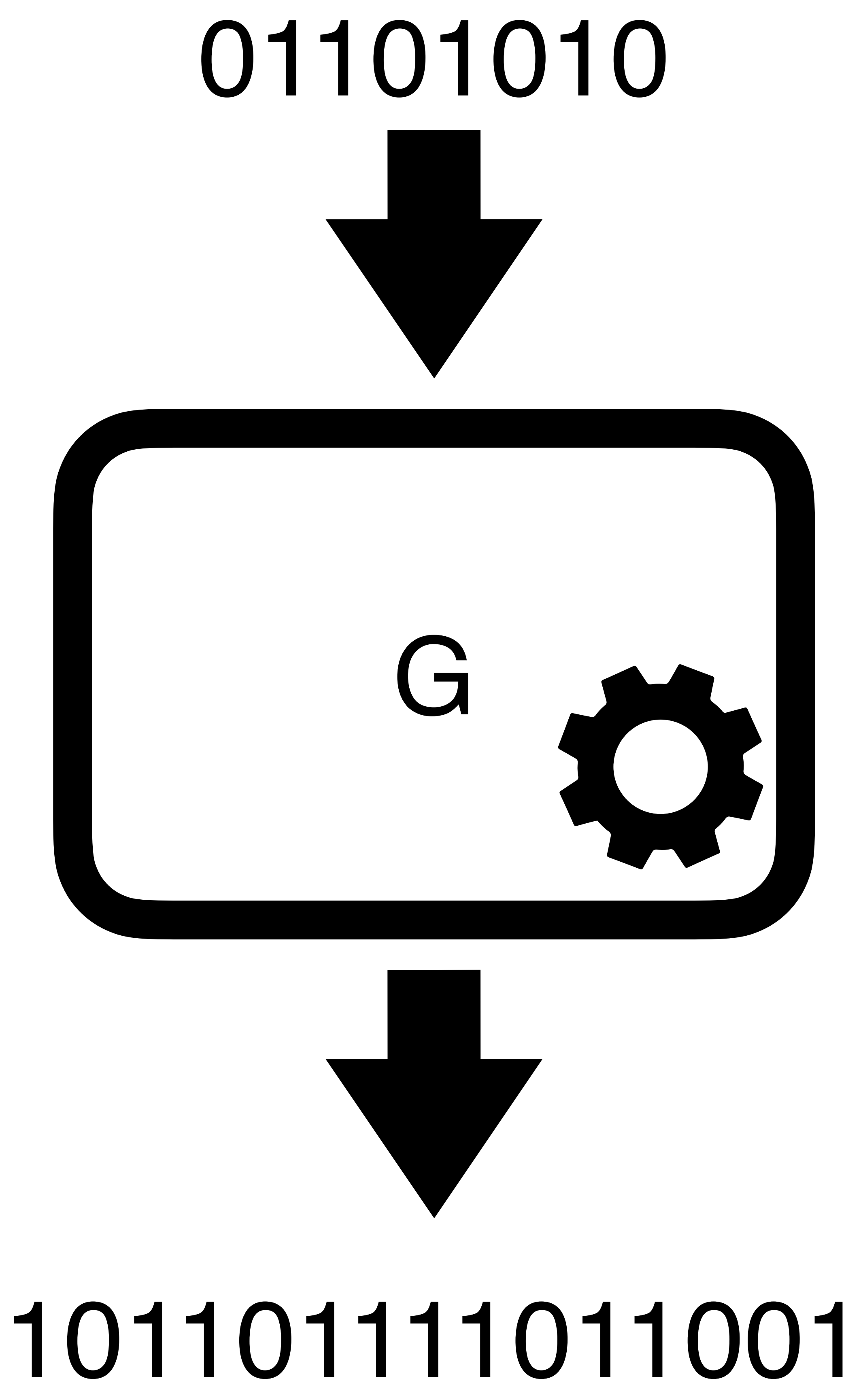


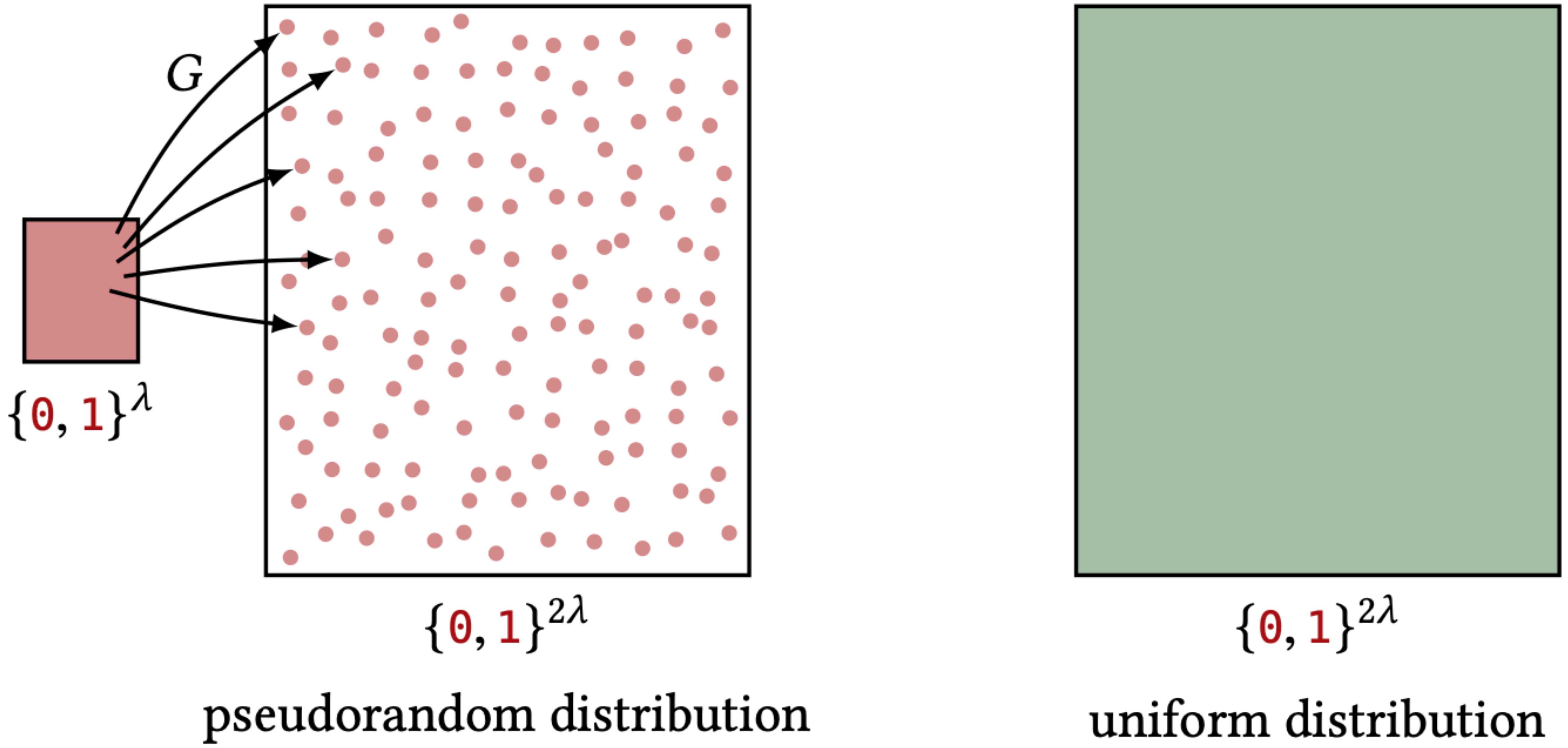
G is a PRG if *no* program can reliably win this game

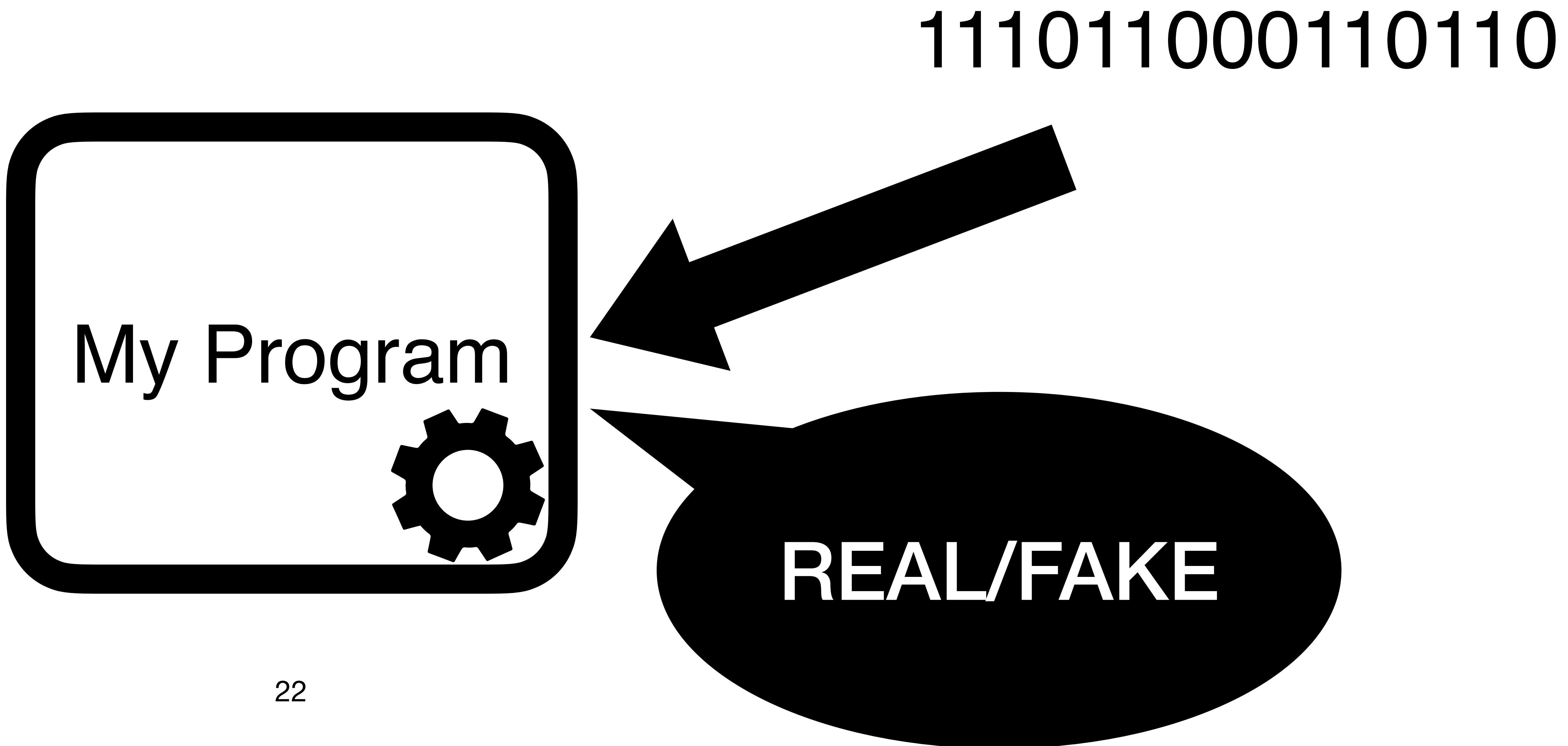


We believe that PRGs exist

If they do, $P \neq NP$

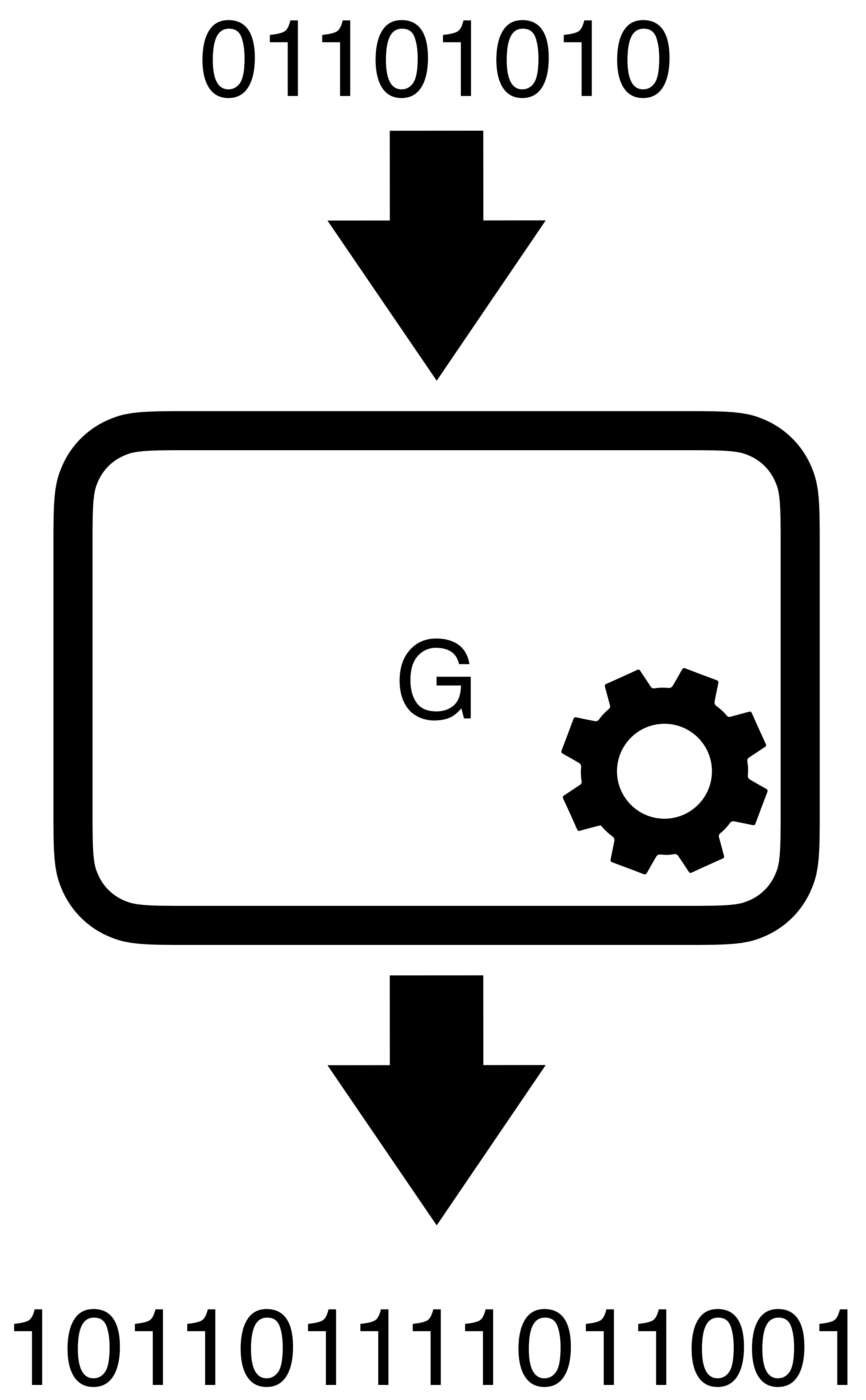




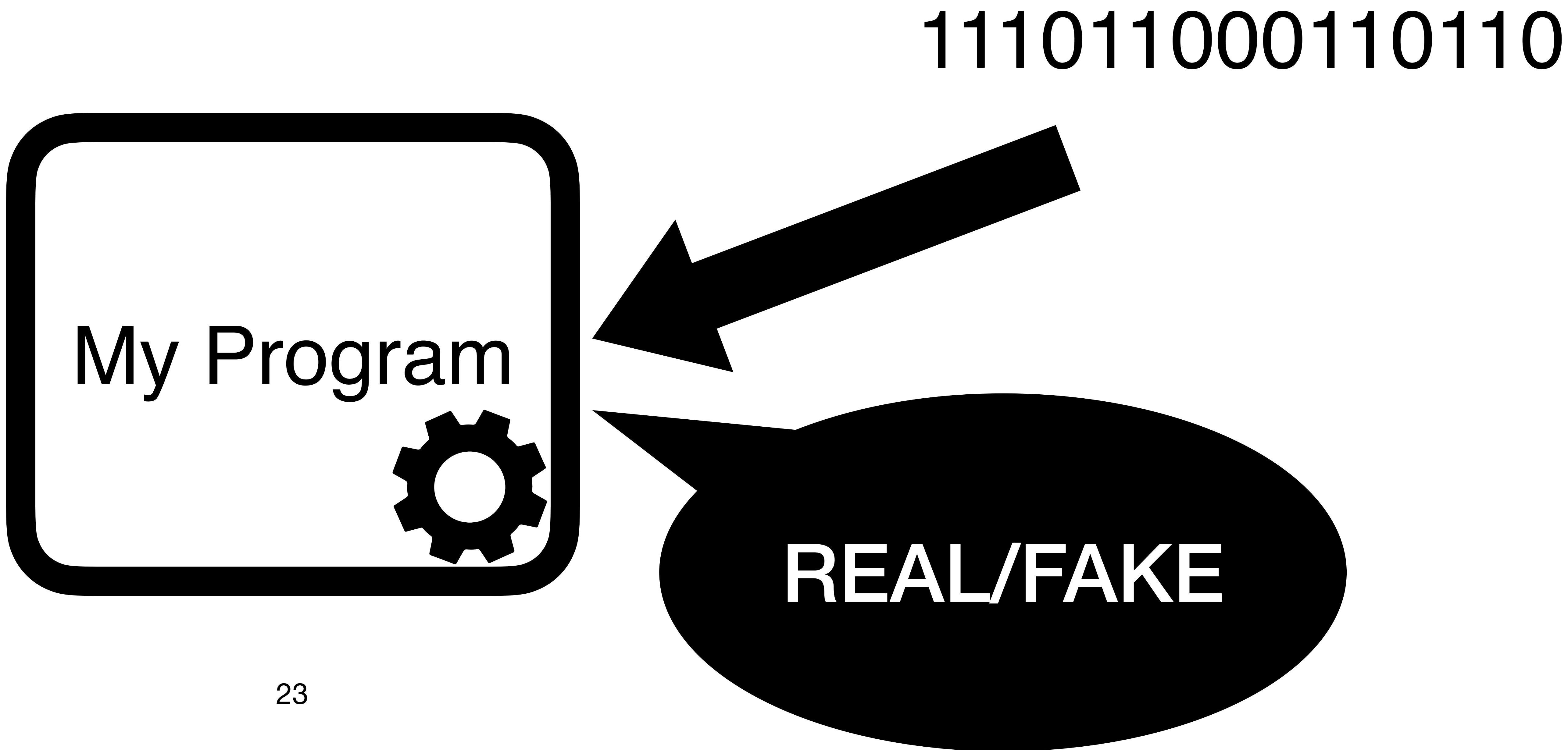


We believe that PRGs exist

If they do, $P \neq NP$



Goal: Make this more precise



We believe that PRGs exist

If they do, $P \neq NP$

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"

Probability Ensemble

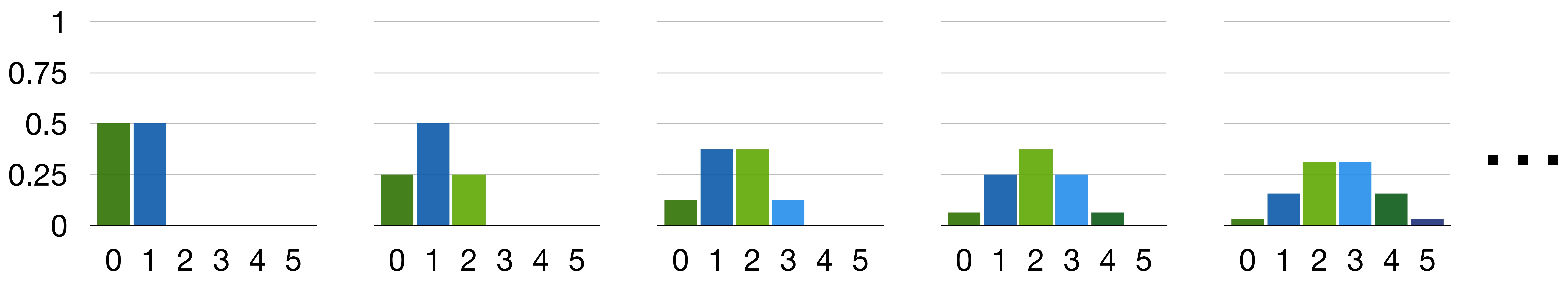
A probability ensemble is a family of probability distributions indexed by the natural numbers.

We typically call this index the ***security parameter,*** λ

Probability Ensemble

A probability ensemble is a family of probability distributions indexed by the natural numbers.

We typically call this index the ***security parameter,*** λ



Sum of outcome of λ coin tosses

Indistinguishability

$$\left\{ s \stackrel{?}{=} 0^\lambda \;\middle|\; s \leftarrow_{\$} \{0,1\}^\lambda \right\}_\lambda \qquad \{\, \mathit{false} \,\}_\lambda$$

These ensembles are “the same”

Indistinguishability

$$\left\{ s \stackrel{?}{=} 0^\lambda \;\middle|\; s \leftarrow_\$ \{0,1\}^\lambda \right\}_\lambda \qquad\qquad \{\ \mathit{false}\ \}_\lambda$$

These ensembles are “the same”

As λ increases, they become harder to tell apart, **very** quickly

Indistinguishability

$$\left\{ s \stackrel{?}{=} 0^\lambda \;\middle|\; s \leftarrow_\$ \{0,1\}^\lambda \right\}_\lambda \qquad \{ \; \mathit{false} \; \}_\lambda$$

These ensembles are "the same"

As λ increases, they become harder to tell apart, **very** quickly

Imagine showing samples of one ensemble to an adversary. Could they guess which was sampled?

Indistinguishability

Let X, Y be two probability ensembles, and let A be an arbitrary (probabilistic) program that outputs 0 or 1. A's **advantage** is as follows:

$$\text{Advantage}_A(\lambda) = \left| \Pr\left[b = 1 \;\middle|\; \begin{array}{l} x \leftarrow_\$ X_\lambda \\ b \leftarrow A(1^\lambda, x) \end{array} \right] - \Pr\left[b = 1 \;\middle|\; \begin{array}{l} y \leftarrow_\$ Y_\lambda \\ b \leftarrow A(1^\lambda, y) \end{array} \right] \right|$$

Indistinguishability

Let X, Y be two probability ensembles, and let A be an arbitrary (probabilistic) program that outputs 0 or 1. A's **advantage** is as follows:

$$\text{Advantage}_A(\lambda) = \left| \Pr\left[b = 1 \;\middle|\; \begin{array}{l} x \leftarrow_\$ X_\lambda \\ b \leftarrow A(1^\lambda, x) \end{array} \right] - \Pr\left[b = 1 \;\middle|\; \begin{array}{l} y \leftarrow_\$ Y_\lambda \\ b \leftarrow A(1^\lambda, y) \end{array} \right] \right|$$

We say that X, Y are **indistinguishable,** written $X \approx Y$ if for every polynomial-time program A:

$\text{Advantage}_A(\lambda)$ is negligible

best strategy is only negligibly better than guessing

Indistinguishability

$$\left\{ s \stackrel{?}{=} 0^\lambda \;\middle|\; s \leftarrow_\$ \{0,1\}^\lambda \right\}_\lambda \qquad\qquad \{\ \mathit{false}\ \}_\lambda$$

These ensembles are "the same"

They are indistinguishable†

† In fact, they are **statistically close**, which is even stronger

PRG security

Let G be a poly-time deterministic algorithm that on an input of length λ outputs a string of length $\lambda + s(\lambda)$.

G is a PRG if $s(\lambda)$ is always positive, and:

$$\left\{ G(k) \;\middle|\; k \leftarrow_\$ \{0,1\}^{\lambda} \right\}_{\lambda} \approx \left\{ r \;\middle|\; r \leftarrow_\$ \{0,1\}^{\lambda+s(\lambda)} \right\}_{\lambda}$$

PRG security

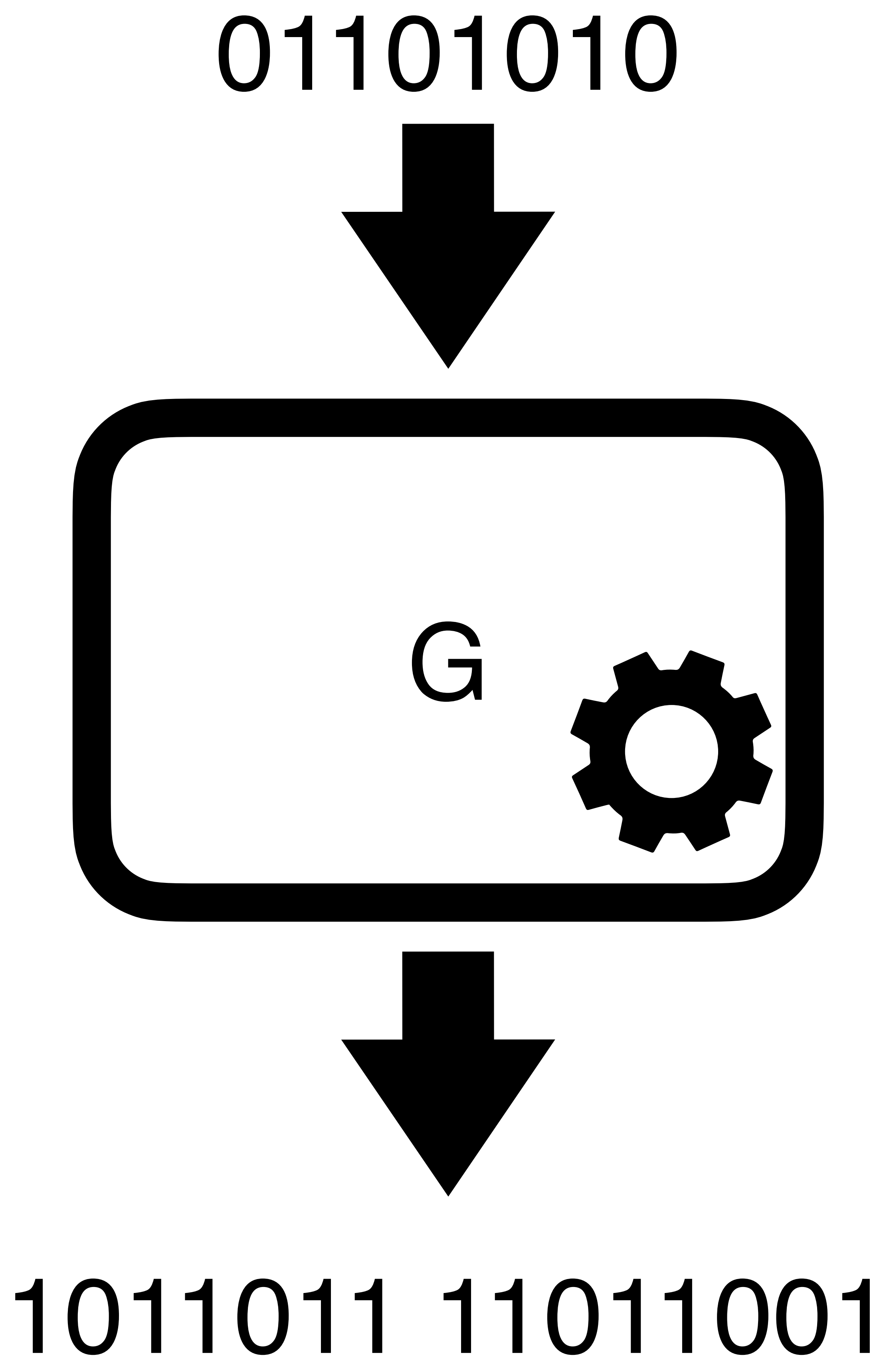
Let G be a poly-time deterministic algorithm that on an input of length λ outputs a string of length $\lambda + s(\lambda)$.

G is a PRG if $s(\lambda)$ is always positive, and:

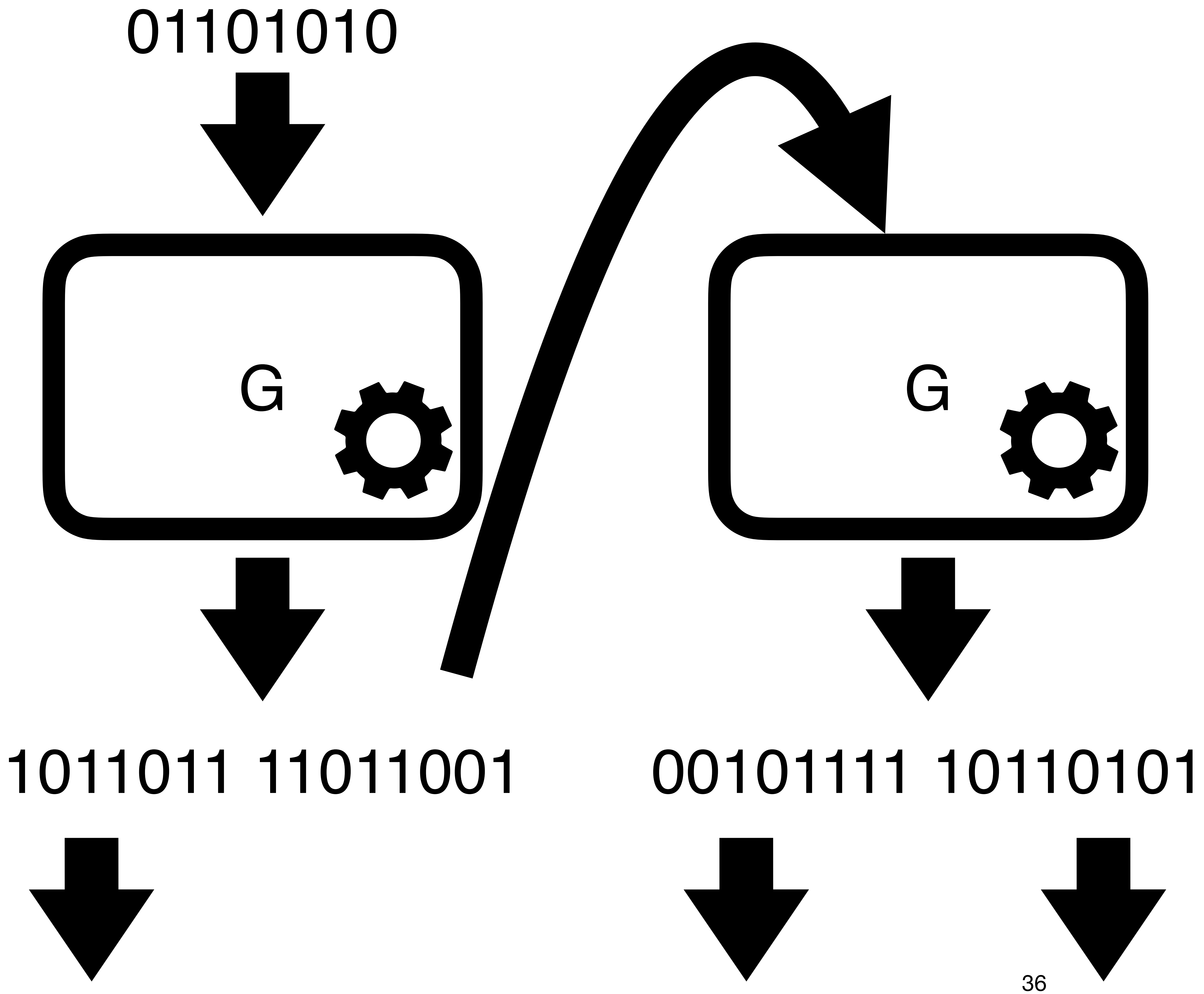
$$\left\{ G(k) \;\middle|\; k \leftarrow_{\$} \{0,1\}^{\lambda} \right\}_{\lambda} \approx \left\{ r \;\middle|\; r \leftarrow_{\$} \{0,1\}^{\lambda+s(\lambda)} \right\}_{\lambda}$$

"If seed k is uniform and hidden, then $G(k)$ looks uniform"

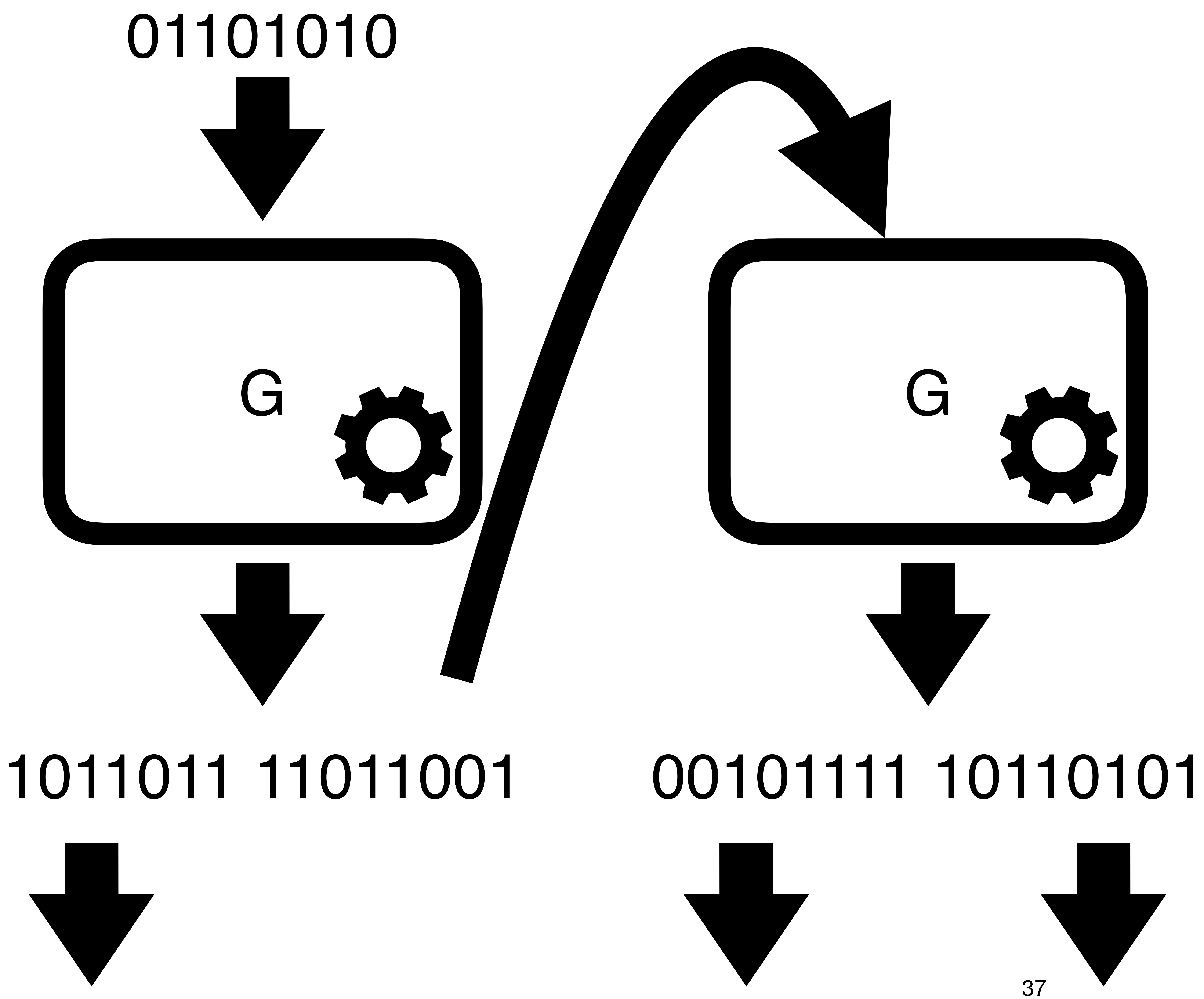
Stretching the output of a PRG



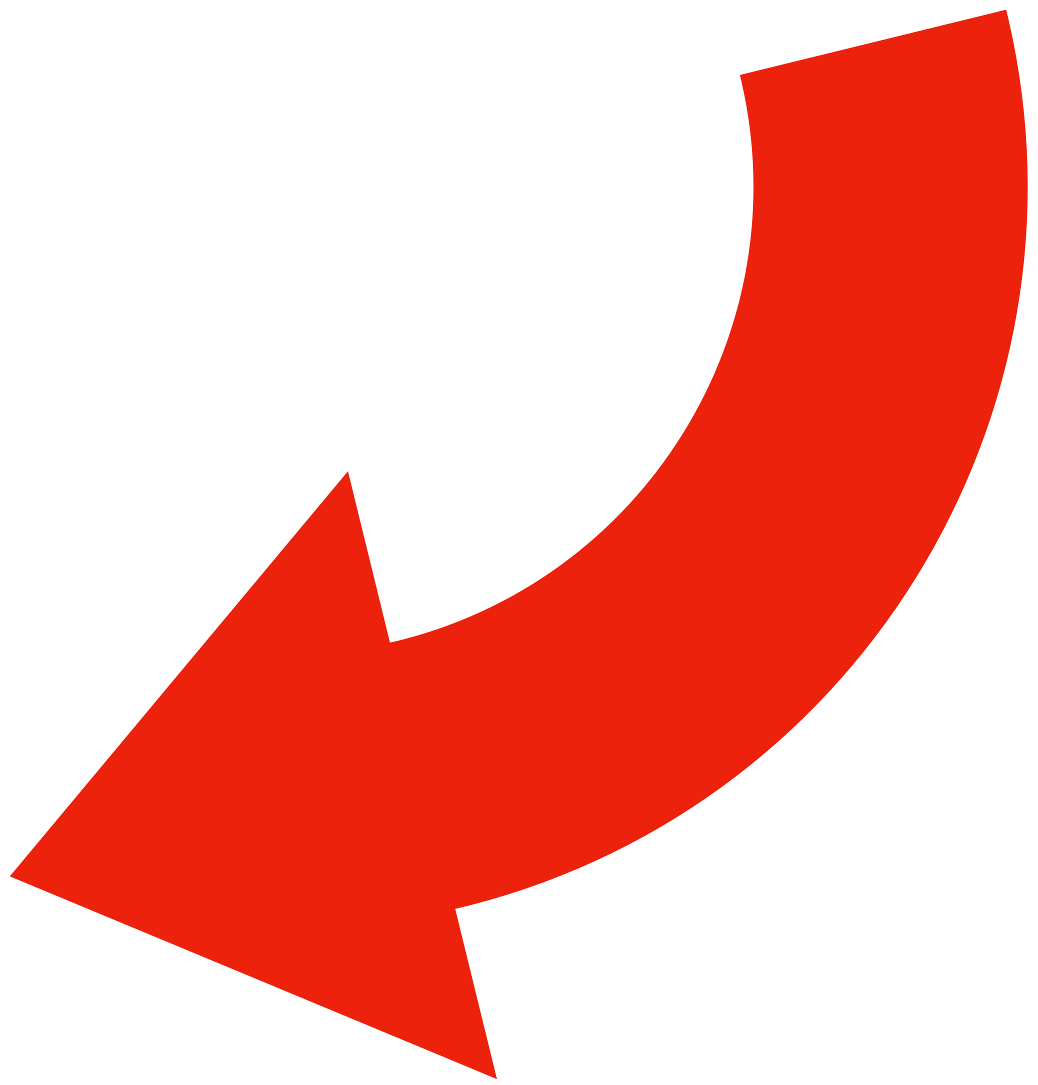
Stretching the output of a PRG



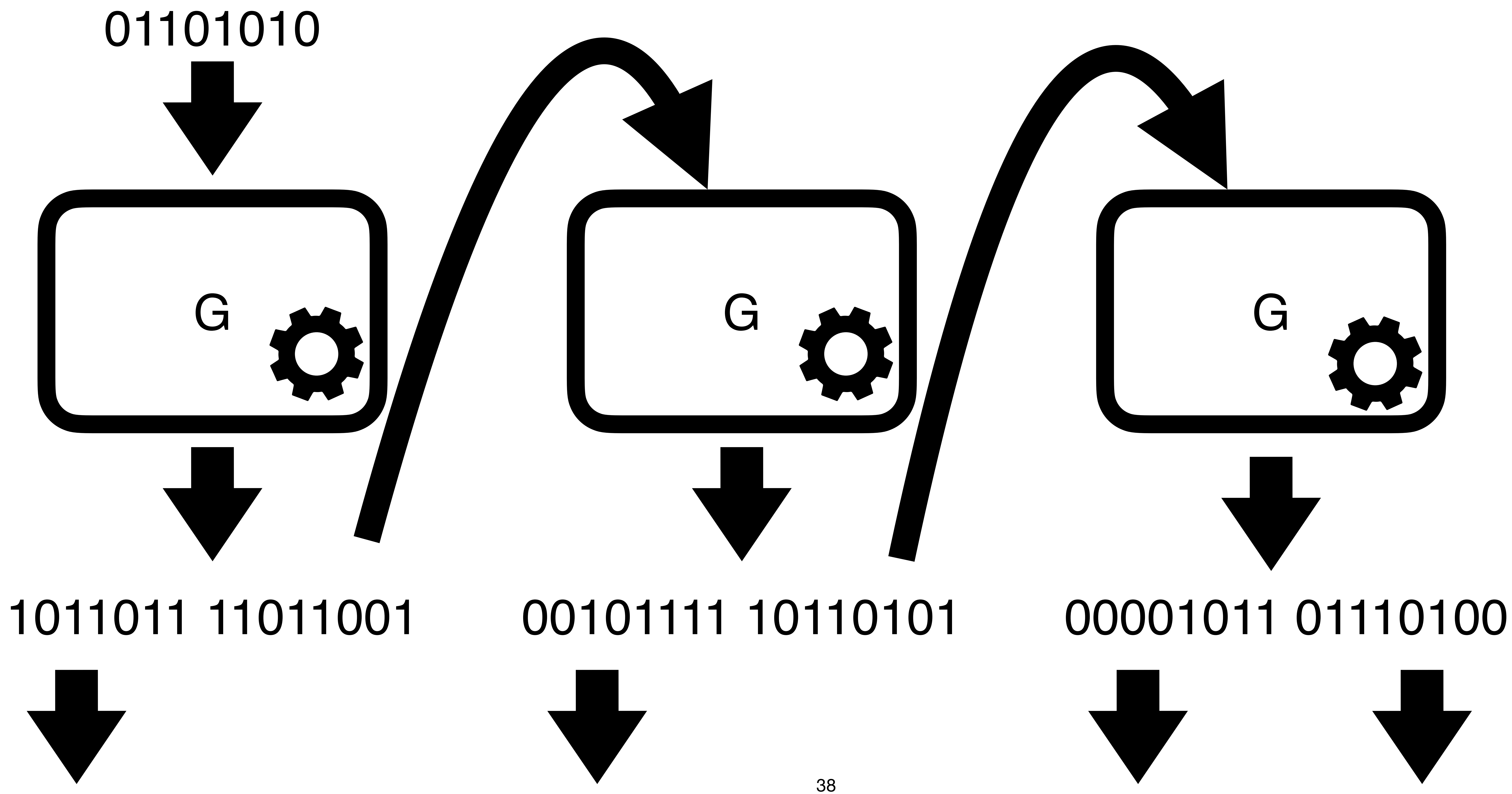
Stretching the output of a PRG

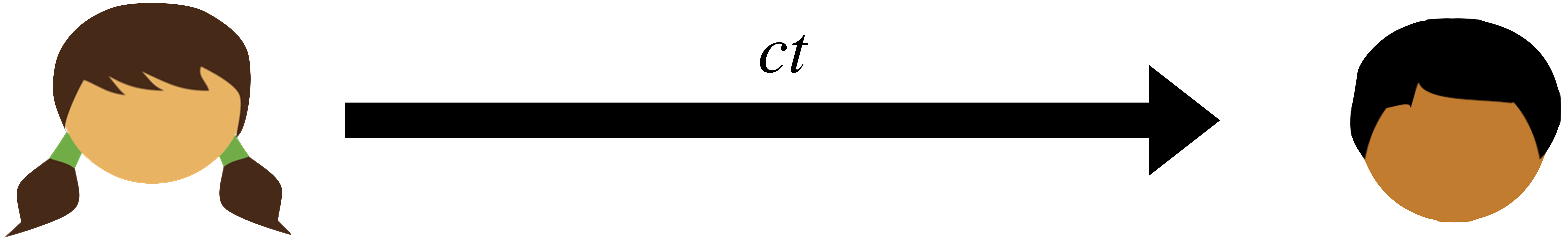


This is a secure PRG



Repeatable any polynomial number of times



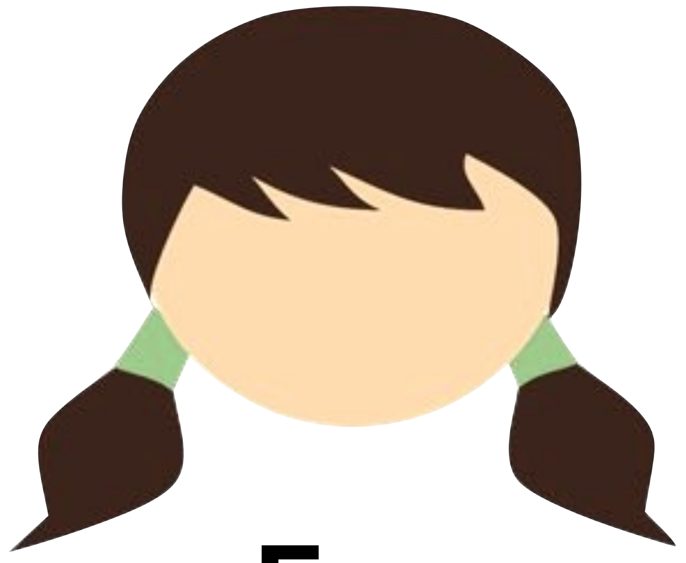


Alice

$m \in \{0,1\}$

$k \leftarrow_{\$} \{0,1\}$

$ct \leftarrow m \oplus k$



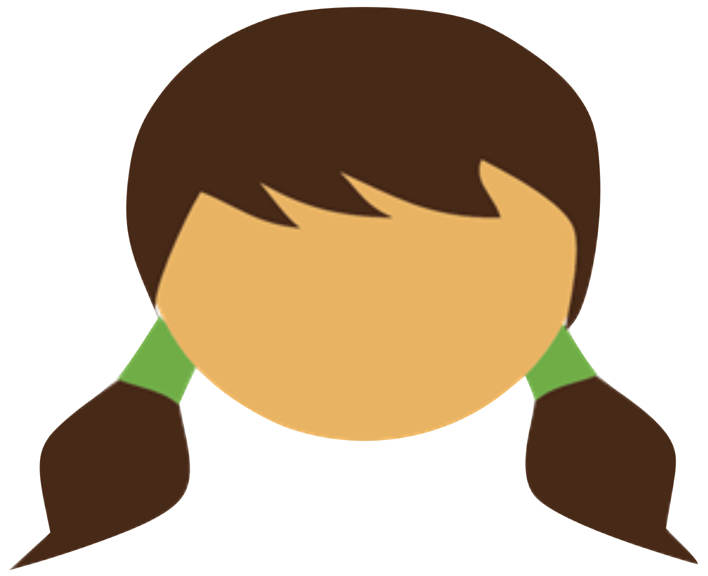
Eve

Bob

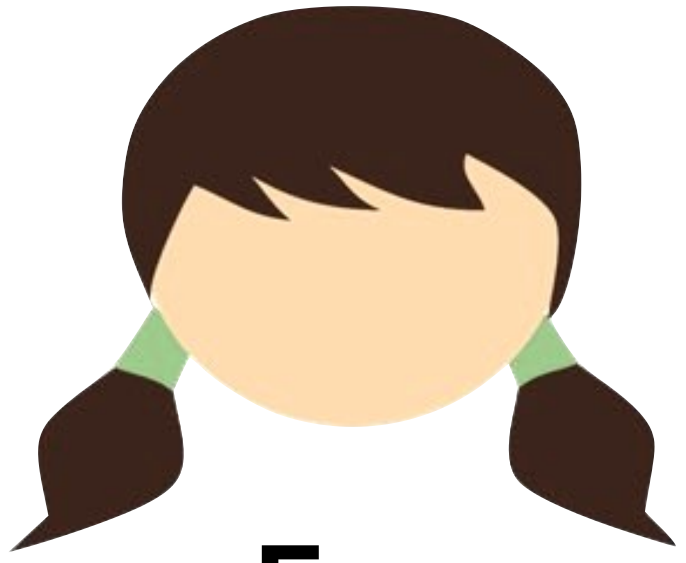
$k \leftarrow_{\$} \{0,1\}$

$m' \leftarrow ct \oplus k$

Question: *what if Alice wants to send more than one bit?*



ct →



Alice

$m \in \{0,1\}$

$k \leftarrow_{\$} \{0,1\}$

$ct \leftarrow m \oplus k$

Eve

Bob

$k \leftarrow_{\$} \{0,1\}$

$m' \leftarrow ct \oplus k$

Question: *what if Alice wants to send more than one bit?*

Answer: *Alice and Bob can exchange a short PRG seed, then expand it (effectively) indefinitely*

Today's objectives

Describe pseudorandomness/pseudorandom generators

Define negligible functions

Introduce indistinguishability