Semi-Honest Security

CS 598 DH

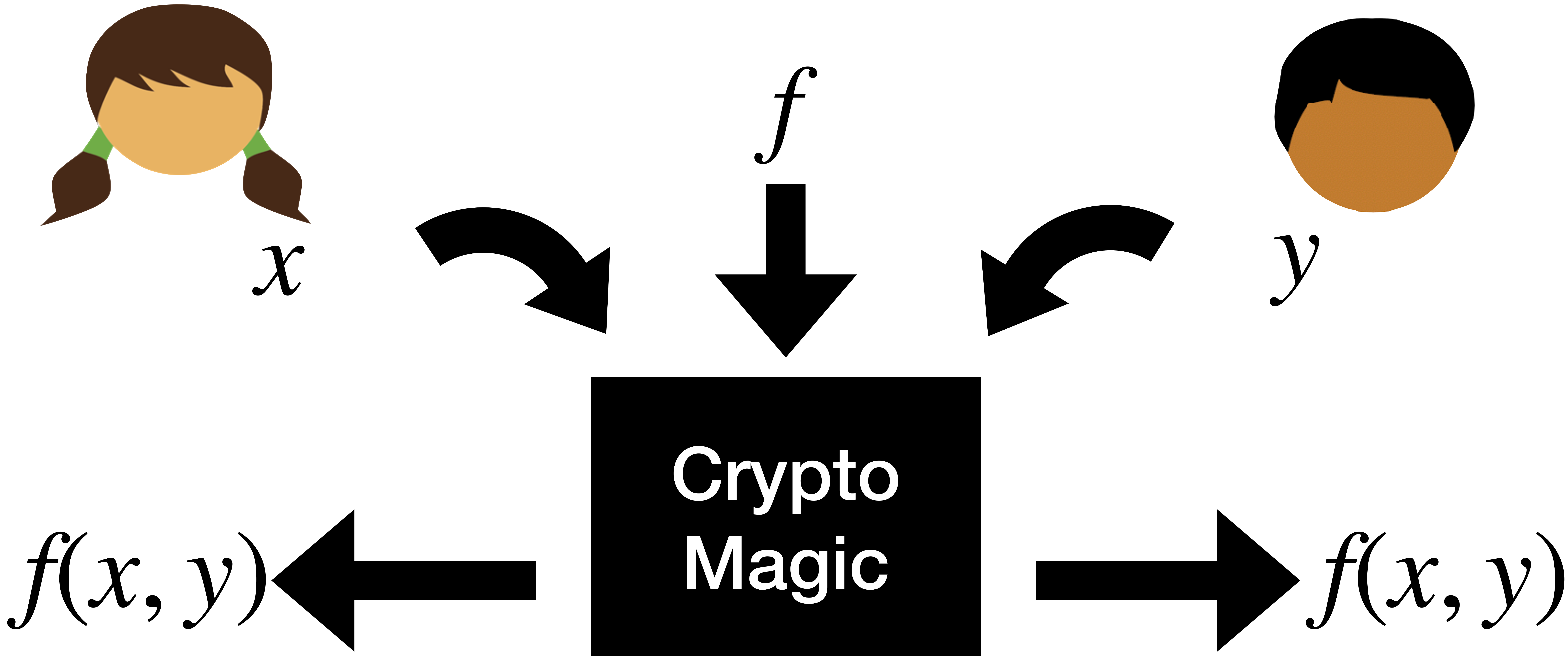
Today’s objectives

Review probability distributions/ensembles

Define negligible functions

Introduce indistinguishability

Formalize semi-honest security

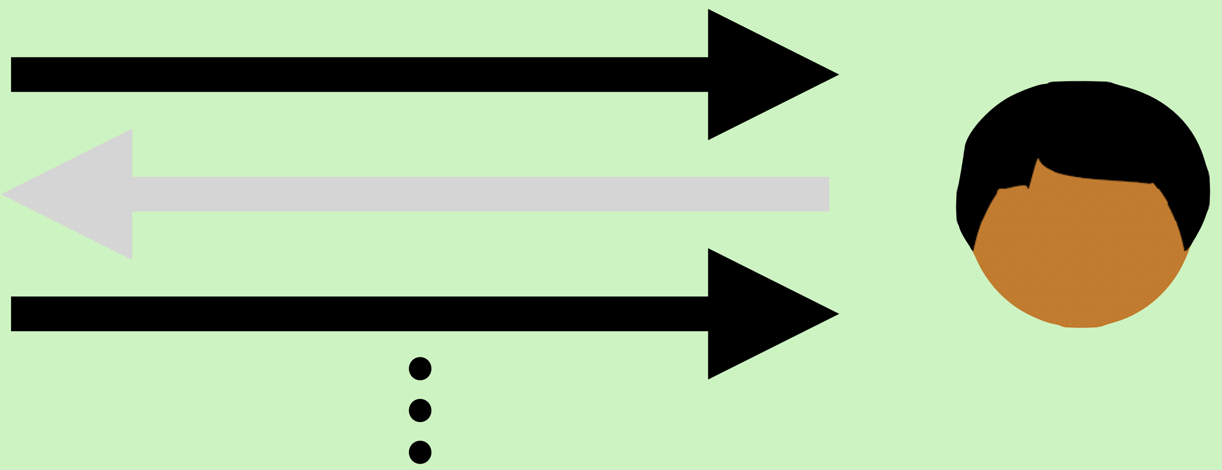


Privacy

Authenticity

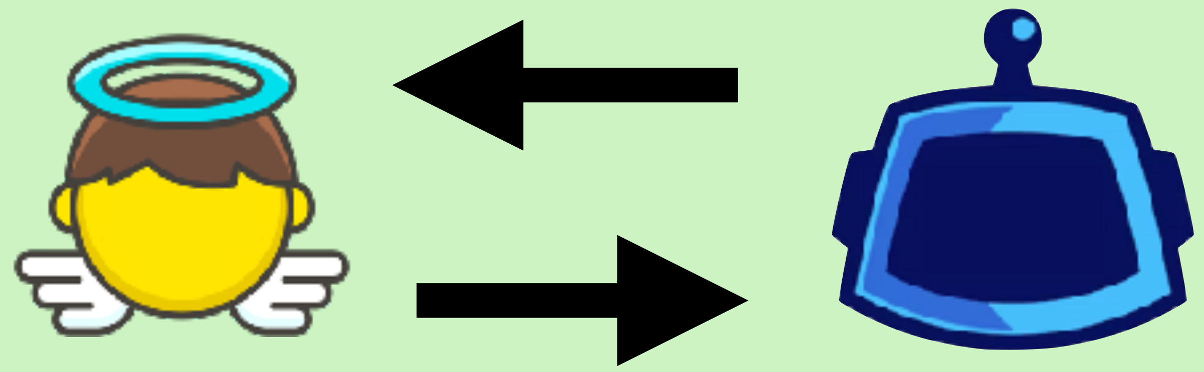
Simulator

Real



$$\text{View}^{\Pi}_{\text{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$$

Ideal



$$\text{Output}^{\text{Sim}}_{\text{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$$

These should "look the same"

```
guess(x : {0,1}ⁿ):
  return false
```

```
secret ←$ {0,1}ⁿ

guess(x : {0,1}ⁿ):
  return x = secret
```

uniformly sample
“Flip n coins at start-up”

```
guess(x : {0,1}^n):
  return false
```

```
secret ←$ {0,1}^n

guess(x : {0,1}^n):
  return x = secret
```

There is a sense in which these two programs are *the same*

uniformly sample
“Flip n coins at start-up”

```
guess(x : {0,1}^n):
  return false
```

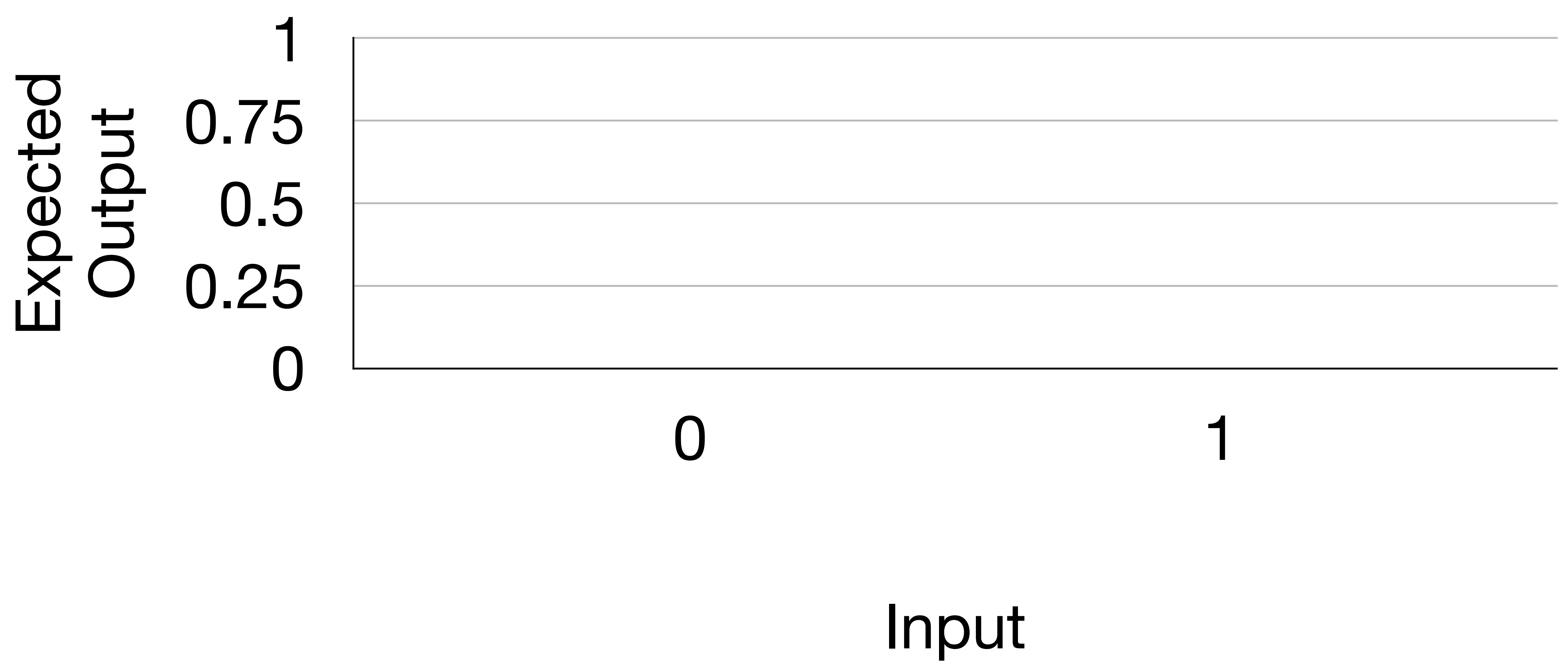
```
secret ←$ {0,1}^n

guess(x : {0,1}^n):
  return x = secret
```

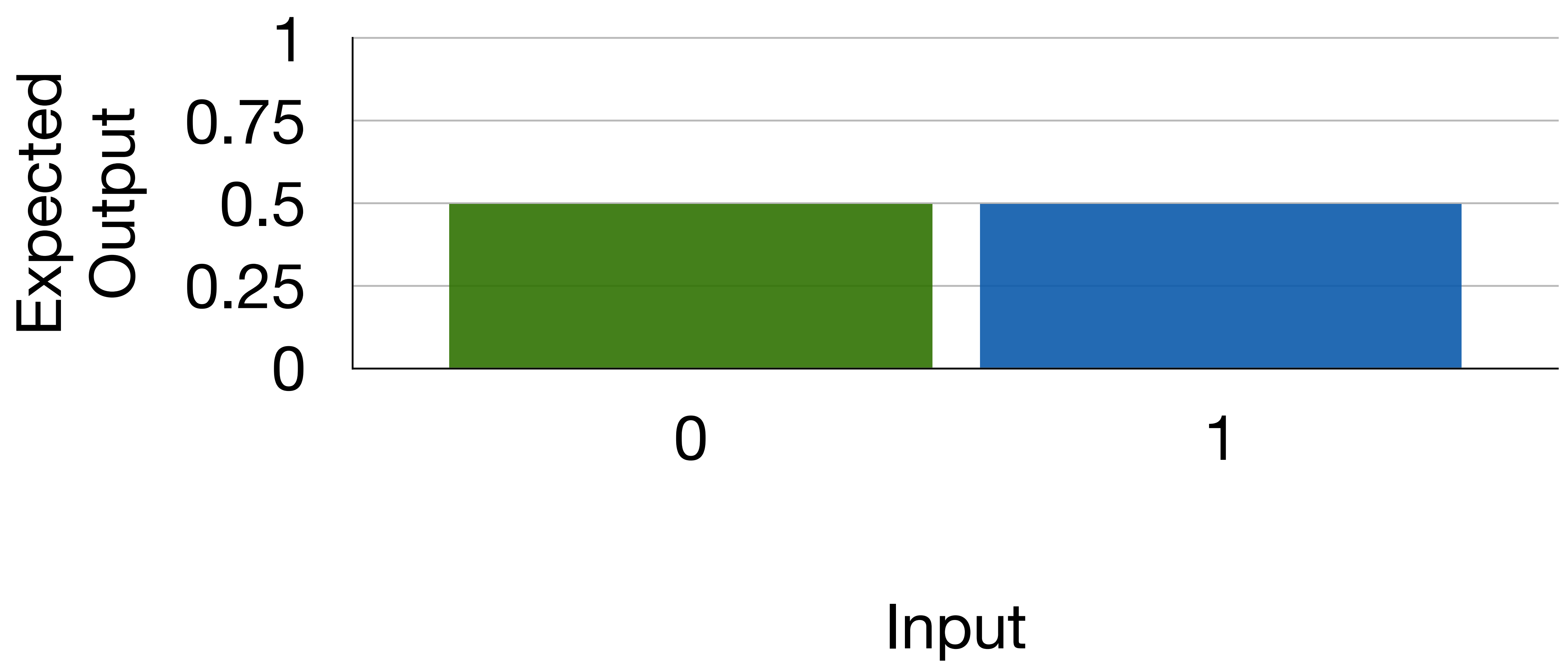
As n increases, the programs become harder and harder to tell apart

```
guess(x : {0,1}^n):
  return 0
```

```
secret ←$ {0,1}^n

guess(x : {0,1}^n):
  return x = secret
```

$n = 1$

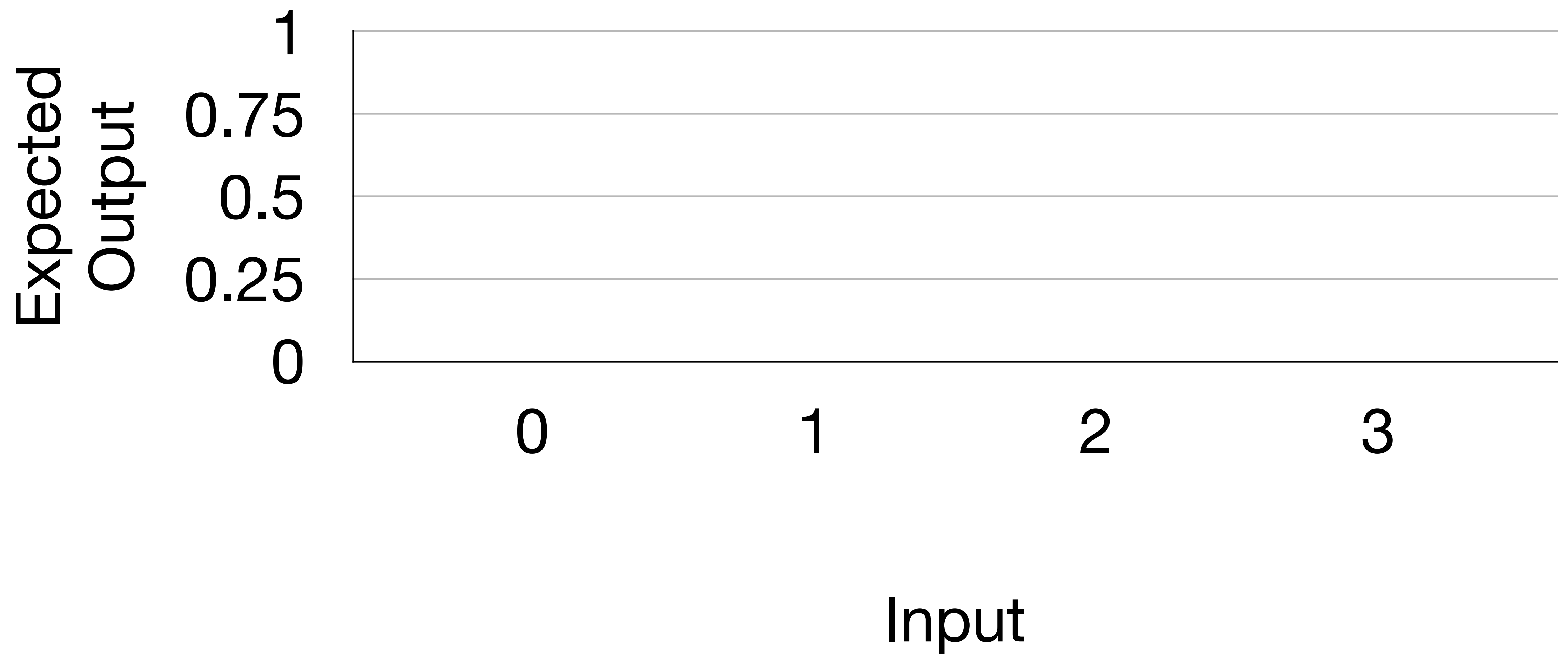


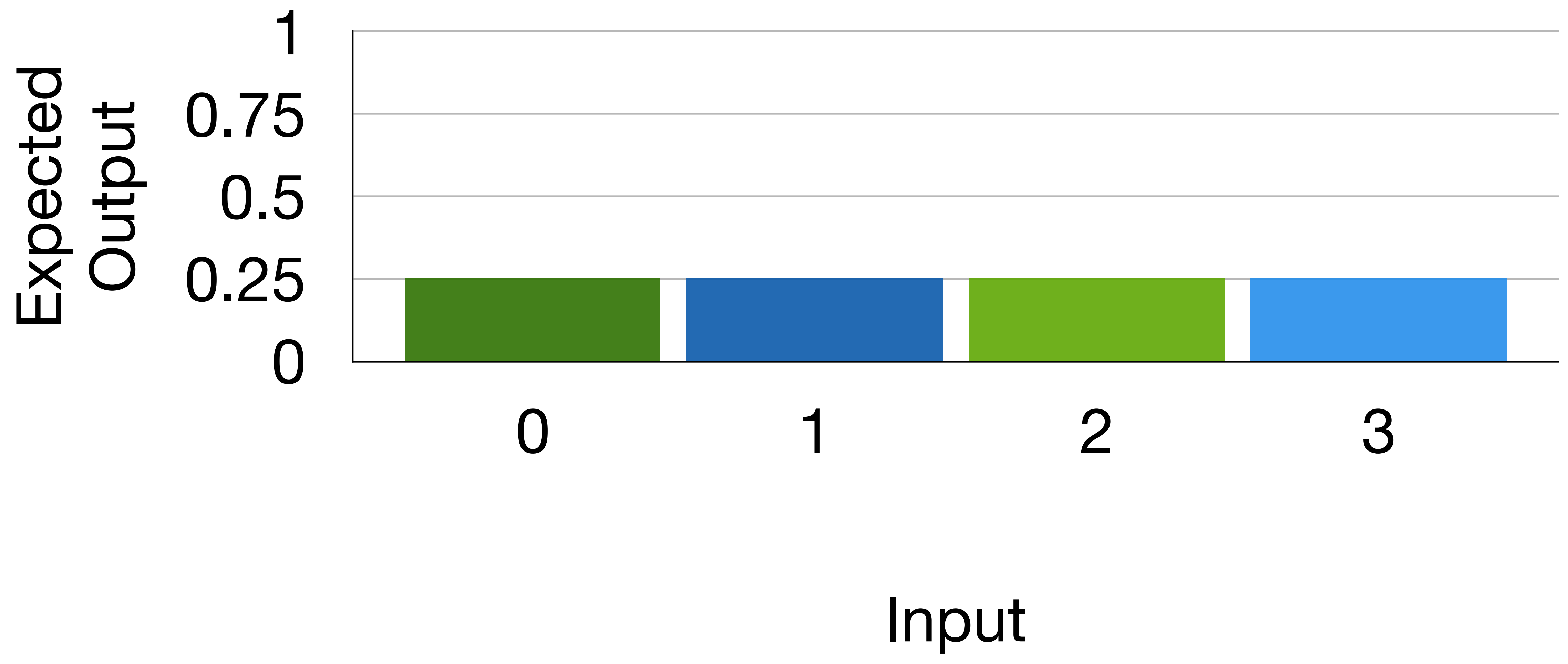


```
guess(x : {0,1}ⁿ):
  return 0
```

```
secret ←$ {0,1}ⁿ

guess(x : {0,1}ⁿ):
  return x = secret
```

$n = 2$

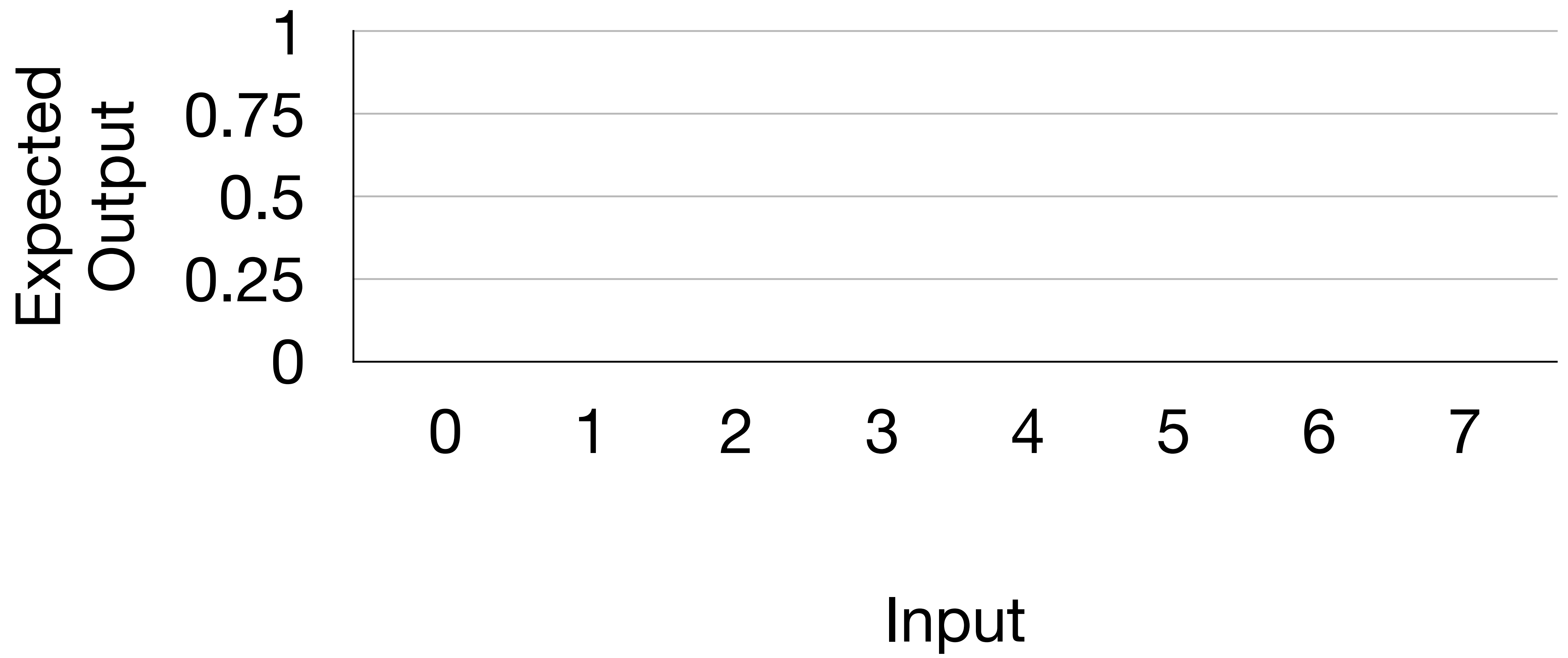


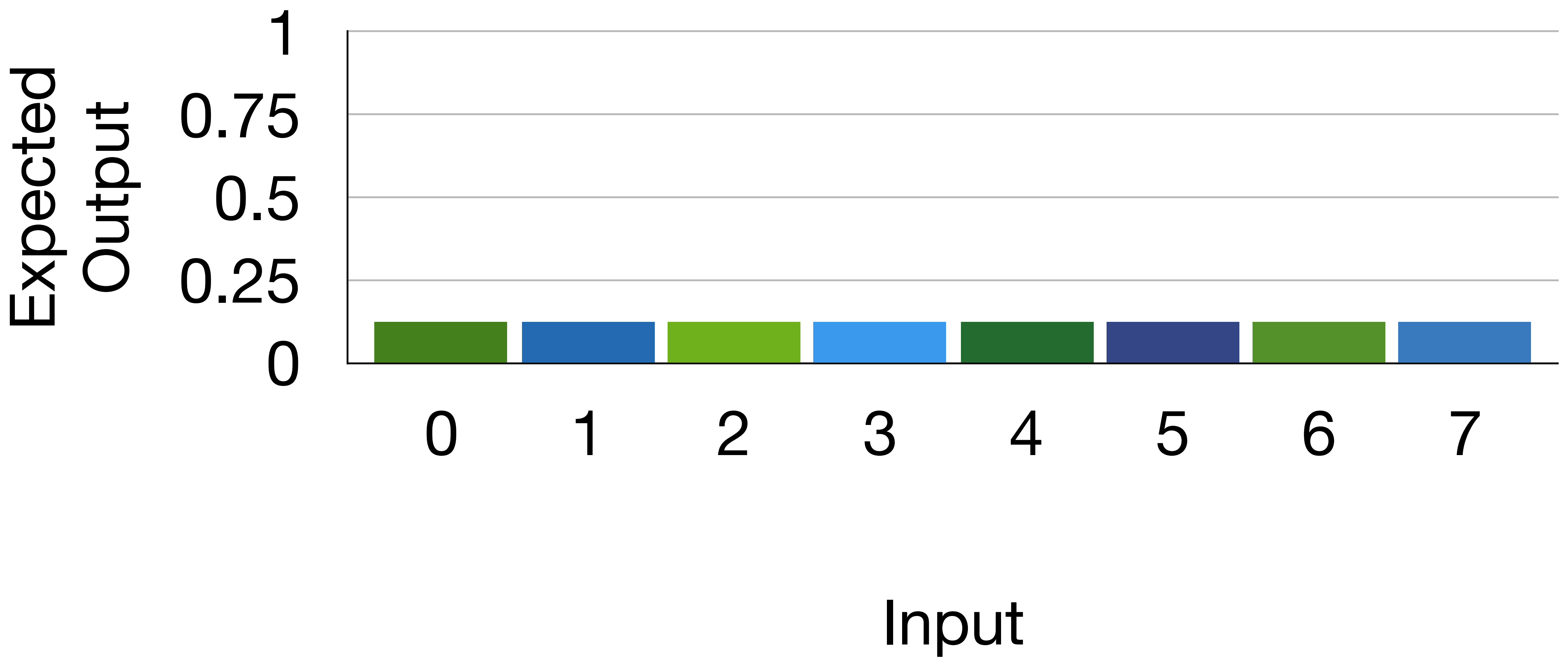


```
guess(x : {0,1}ⁿ):
  return 0
```

```
secret ←$ {0,1}ⁿ

guess(x : {0,1}ⁿ):
  return x = secret
```

$n = 3$

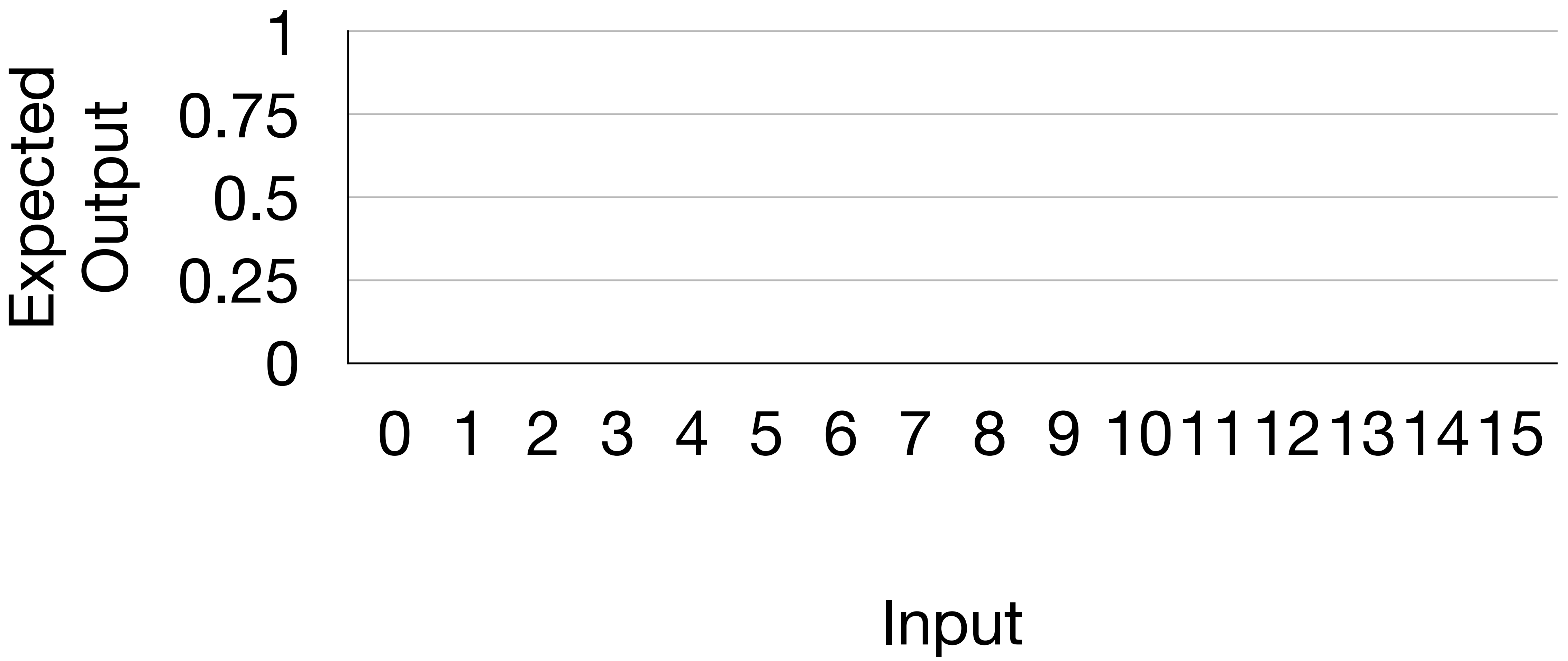


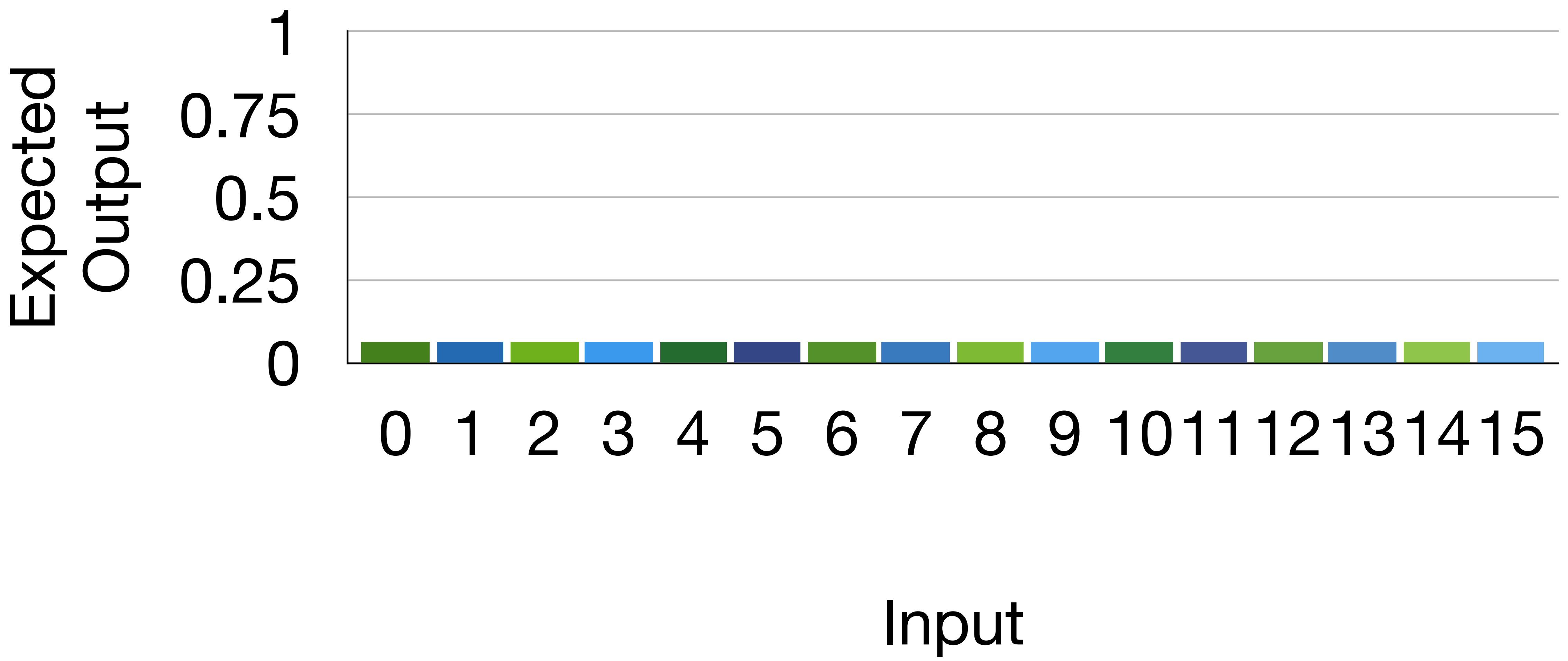


```
guess(x : {0,1}ⁿ):
  return 0
```

```
secret ←$ {0,1}ⁿ

guess(x : {0,1}ⁿ):
  return x = secret
```

$n = 4$





```
guess(x : {0,1}^n):
  return 0
```

```
secret ←$ {0,1}^n

guess(x : {0,1}^n):
  return x = secret
```

A (randomized) program can be viewed as the description of some distribution

Some programs that look very different can describe very similar distributions

(Discrete) Probability Distribution

The probability distribution *associated with a random variable* X *is a function mapping input* x *to the probability that* X *takes value* x

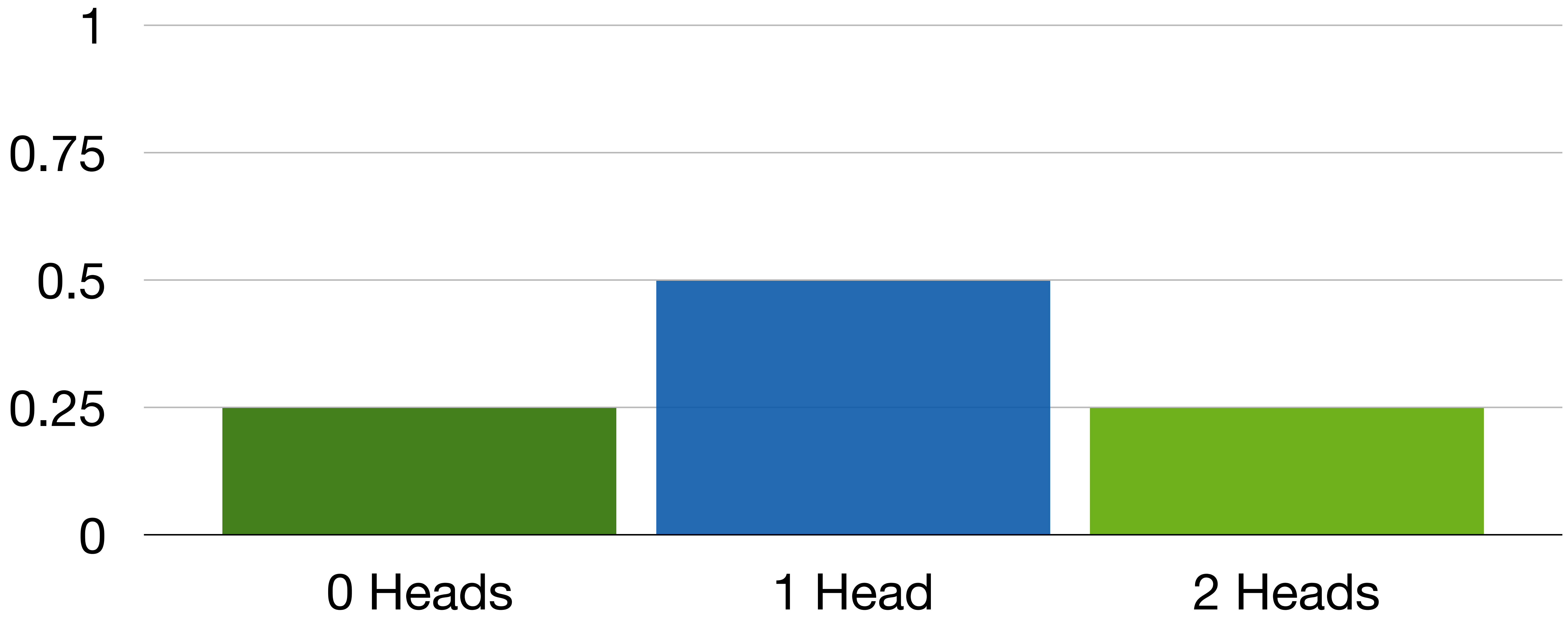
(Discrete) Uniform Distribution

A probability distribution where each outcome is equally likely.

(Discrete) Probability Distribution

The probability distribution *associated with a random variable* X *is a function mapping input* x *to the probability that* X *takes value* x

Flip two fair coins



Probability Ensemble

A Probability Ensemble *is a family of random variables, indexed by a natural number*

$$X = \{\ X_n\ \}_{n \in \mathbb{N}}$$

Probability Ensemble

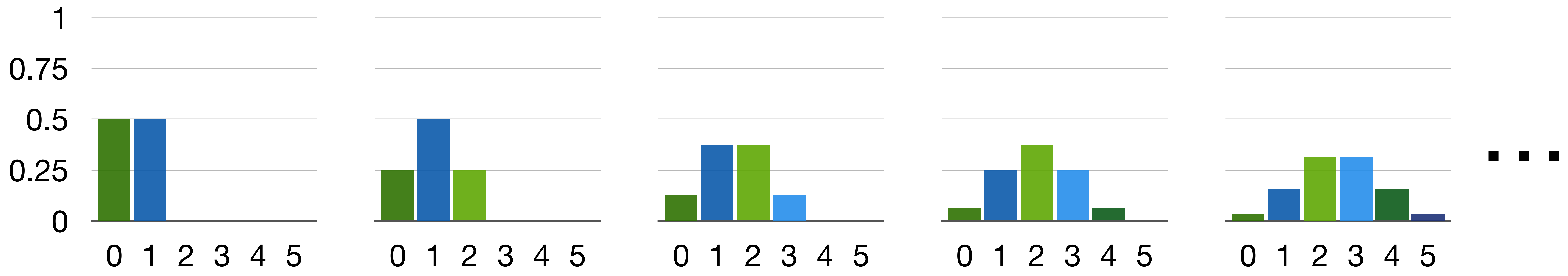
A Probability Ensemble *is a family of random variables, indexed by a natural number*

$$X = \{ \ X_n \ \}_{n \in \mathbb{N}}$$

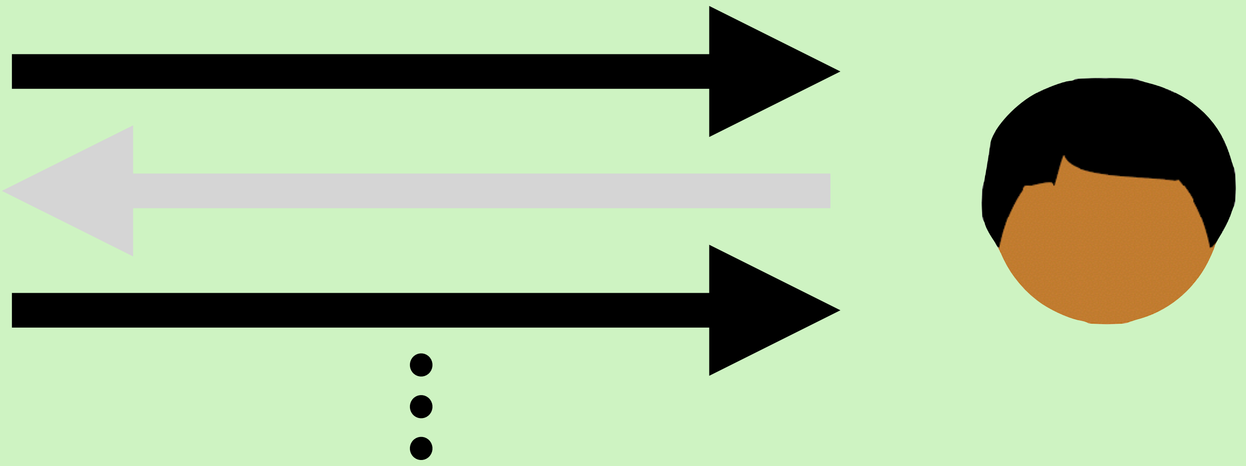
Hint: asymptotic behavior. How does this random variable change as we increase n?

Probability Ensemble

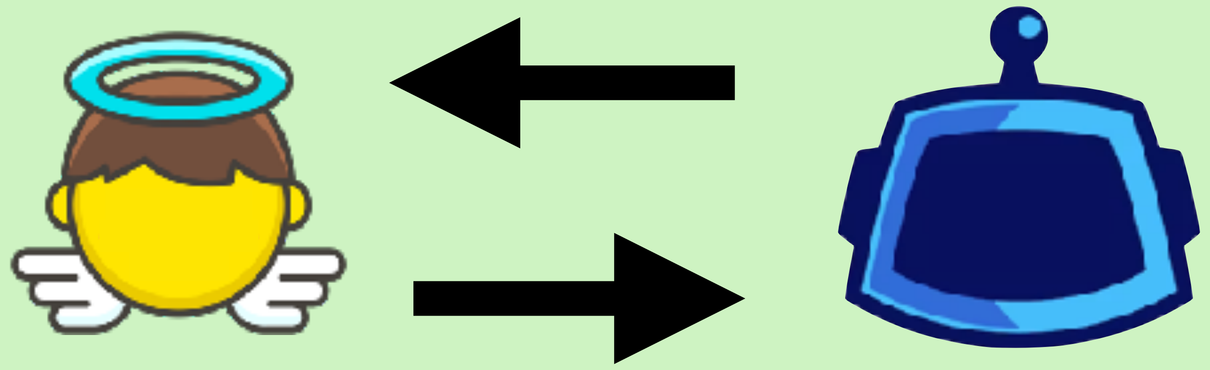
A Probability Ensemble *is a family of random variables, indexed by a natural number*



Number of heads as we increase the number of coin flips



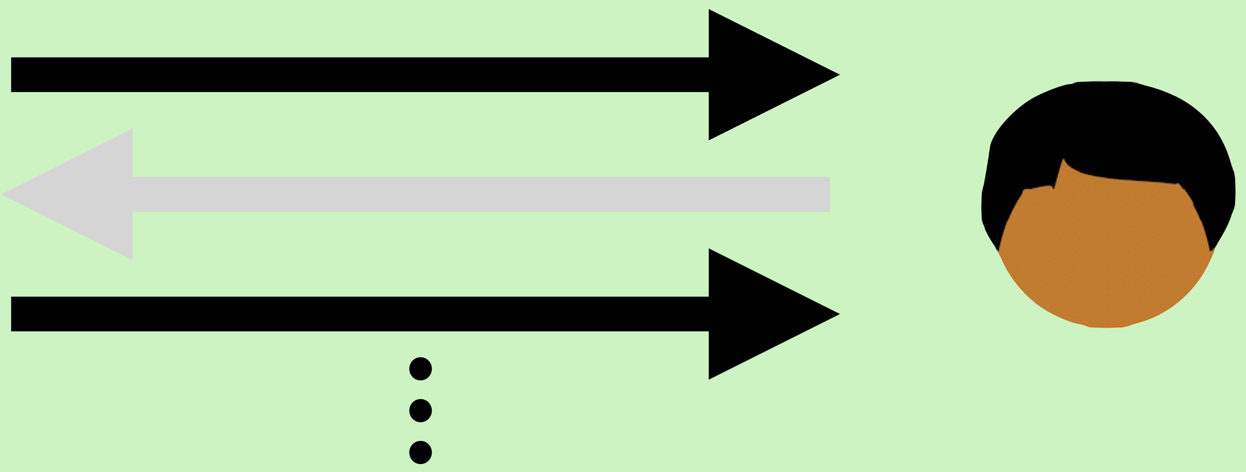
$$\mathrm{View}^{\Pi}_{\mathrm{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$$



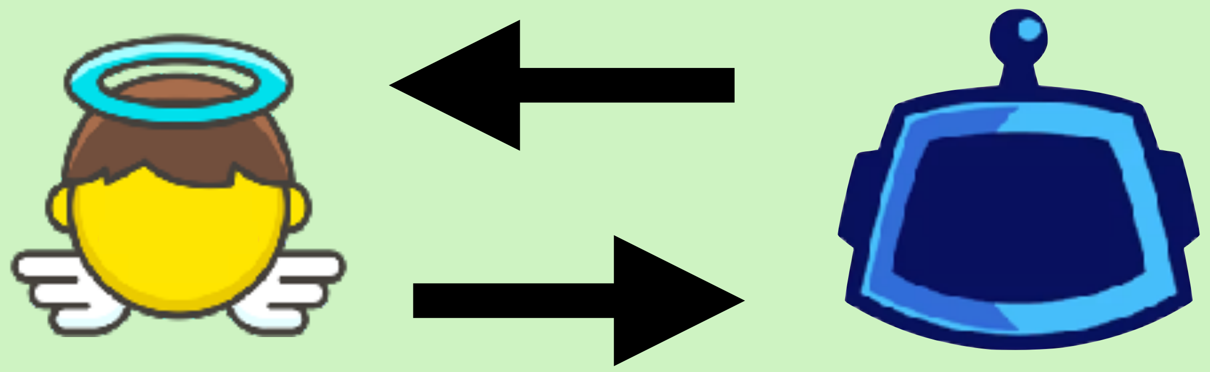
$$\mathrm{Output}^{\mathrm{Sim}}_{\mathrm{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$$

These ensembles are hard to tell apart

“No efficient algorithm can tell these two things apart”



$\text{View}^{\Pi}_{\text{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$



$\text{Output}^{\text{Sim}}_{\text{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$

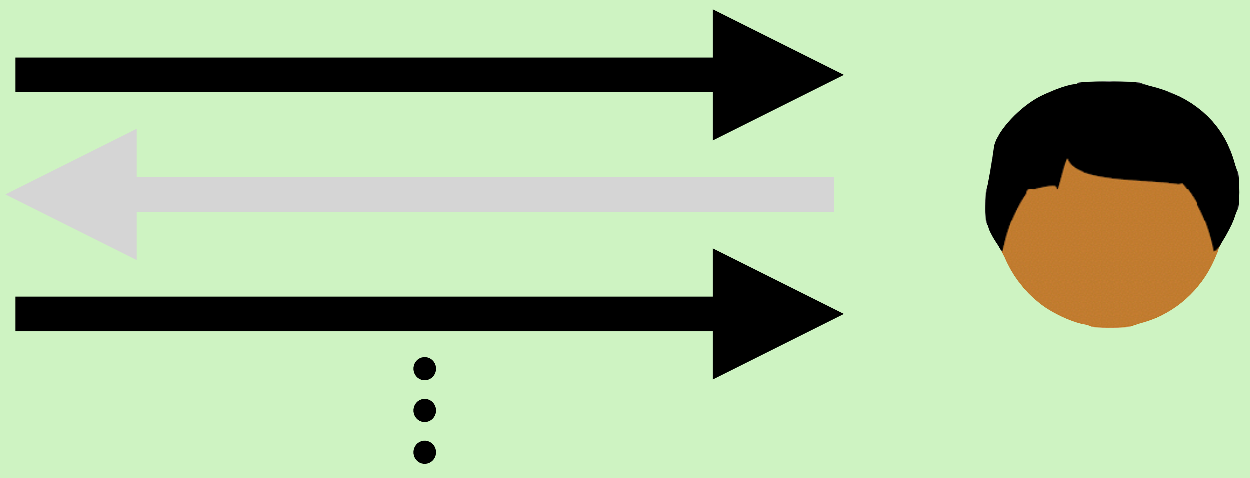
Three notions of “hard to tell apart”

Identically distributed

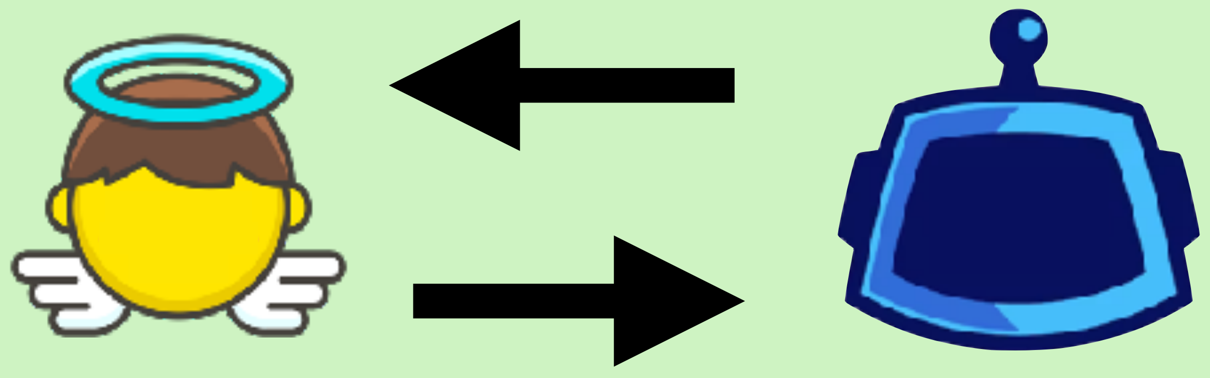
Statistically close

Indistinguishable

"No efficient algorithm can tell these two things apart"



$\mathrm{View}^{\Pi}_{\mathrm{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$



$\mathrm{Output}^{\mathrm{Sim}}_{\mathrm{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$

Three notions of "hard to tell apart"

Identically distributed

Statistically close — As we increase a parameter, the distributions **quickly** become close together.

Indistinguishable — As we increase a parameter, it **quickly** becomes difficult for programs to tell the distributions apart.

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"

Negligible Function

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

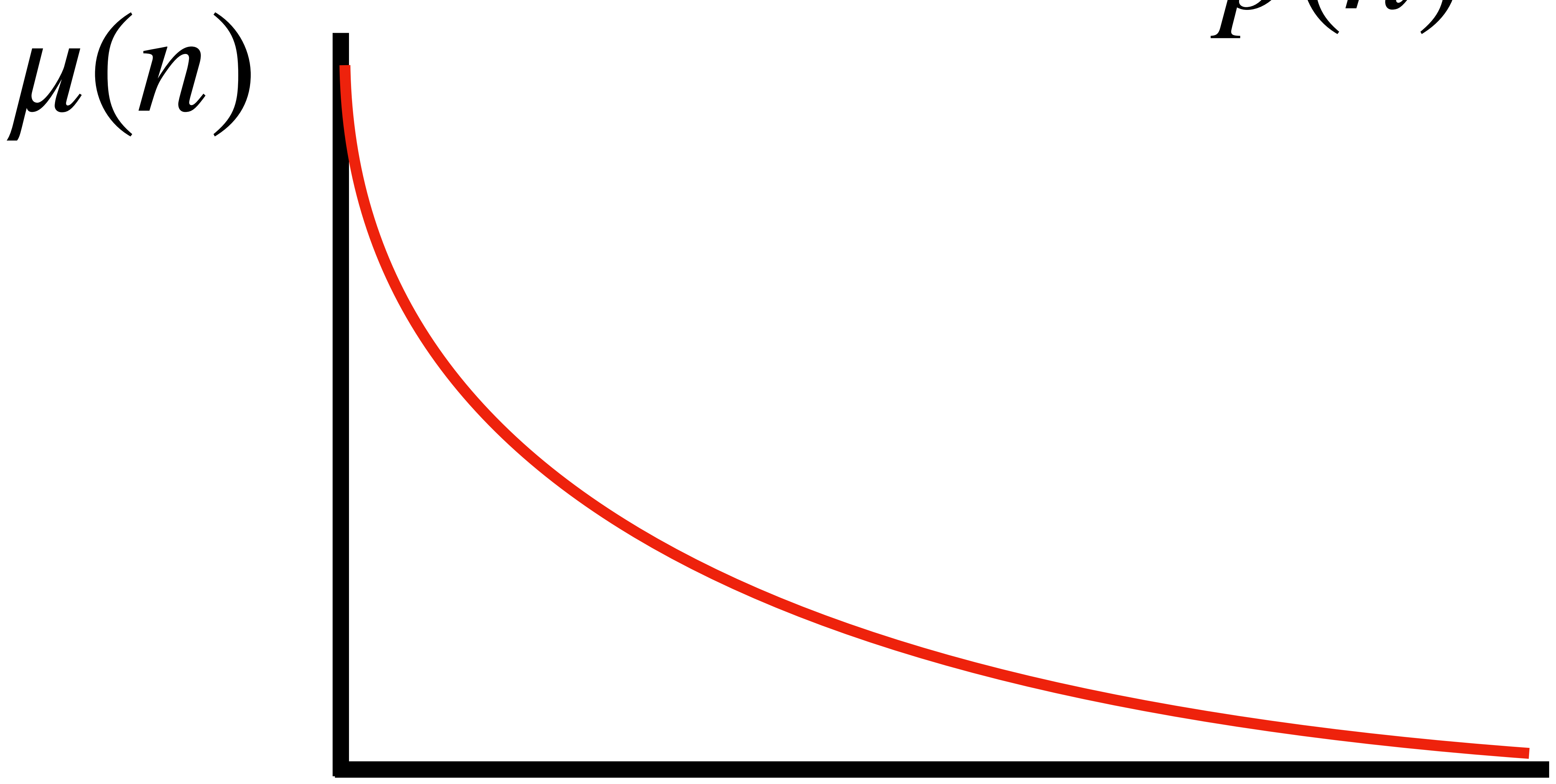
$$\mu(n) < \frac{1}{p(n)}$$

Canonical example: $\mu(n) = \frac{1}{2^n}$

Negligible Function

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

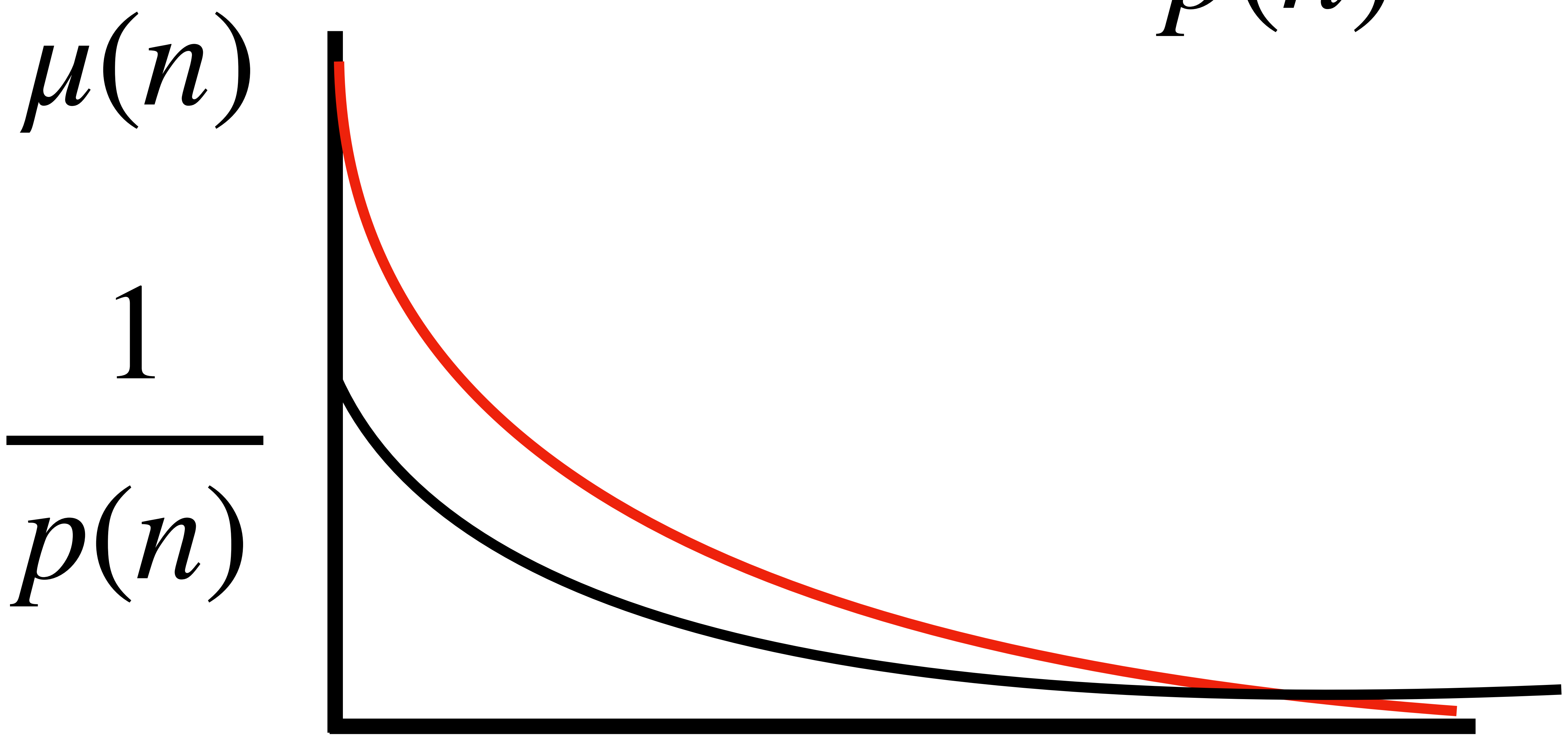
$$\mu(n) < \frac{1}{p(n)}$$



Negligible Function

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

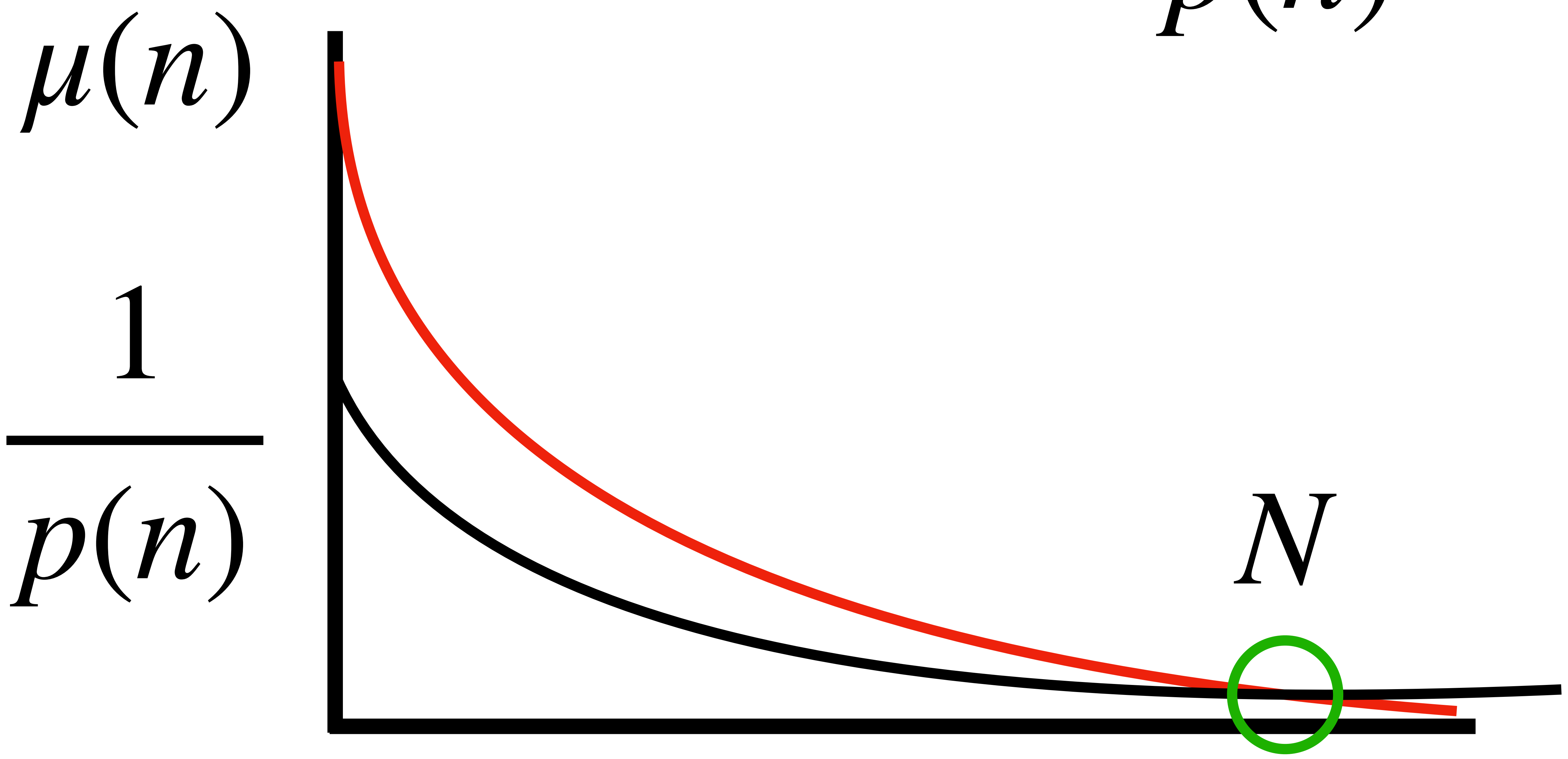
$$\mu(n) < \frac{1}{p(n)}$$



Negligible Function

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

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



Statistically Close

Statistical Distance

$$\Delta(X, Y) = \frac{1}{2} \sum_{\alpha \in \mathrm{Domain}} \left| \Pr[X = \alpha] - \Pr[Y = \alpha] \right|$$

Statistically Close

Statistical Distance

$$\Delta(X, Y) = \frac{1}{2} \sum_{\alpha \in \mathrm{Domain}} \left| \Pr[X = \alpha] - \Pr[Y = \alpha] \right|$$

Ensembles $\{\ X_n\ \}$ and $\{\ Y_n\ \}$ are **statistically close** if the following is a negligible function:

$$f(n) = \Delta(X_n, Y_n)$$

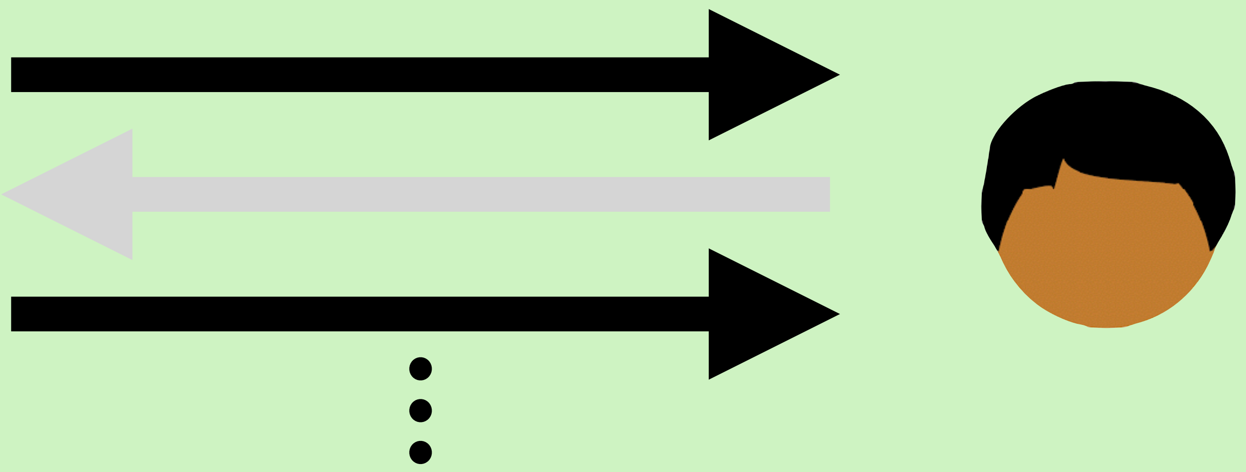
Indistinguishability

Let X, Y *be ensembles.*

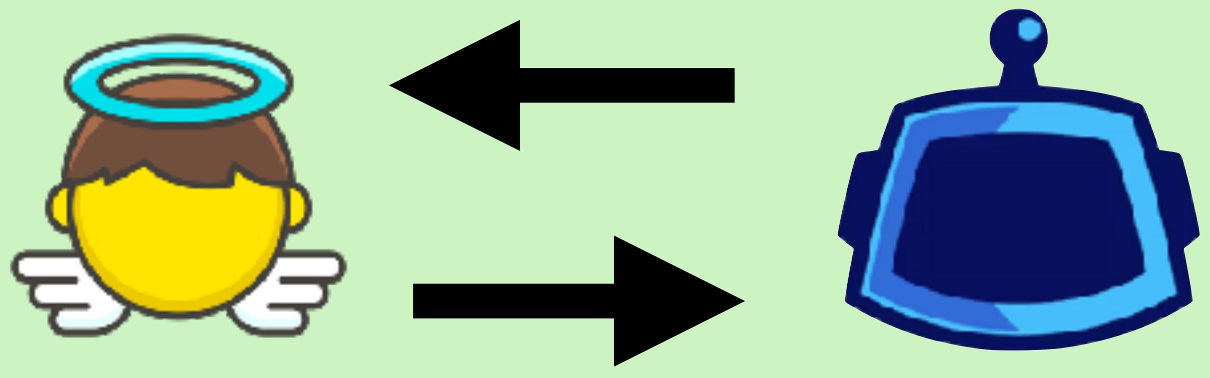
We say that X *and* Y *are* computationally indistinguishable if for every (non-uniform) polynomial-time program $\mathscr{D}$, the following function is negligible:

$$\delta(n) = \left| \left(\Pr_{x \leftarrow X_n} \left[\ \mathscr{D}(x) = 1 \ \right] \right) - \left(\Pr_{y \leftarrow Y_n} \left[\ \mathscr{D}(y) = 1 \ \right] \right) \right|$$

“No efficient algorithm can tell these two things apart”



$\text{View}^{\Pi}_{\text{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$



$\text{Output}^{\text{Sim}}_{\text{Bob}}(x, y) = \{\ y, m_0, m_1, \ldots\ \}$

Three notions of “hard to tell apart”

$X \equiv Y$ Identically distributed

$X \approx Y$ Statistically close — As we increase a parameter, the distributions **quickly** become close together.

$X \overset{c}{=} Y$ Indistinguishable — As we increase a parameter, it **quickly** becomes difficult for programs to tell the distributions apart.

uniformly sample
“Flip n coins at start-up”

```
guess(x : {0,1}ⁿ):
  return false
```

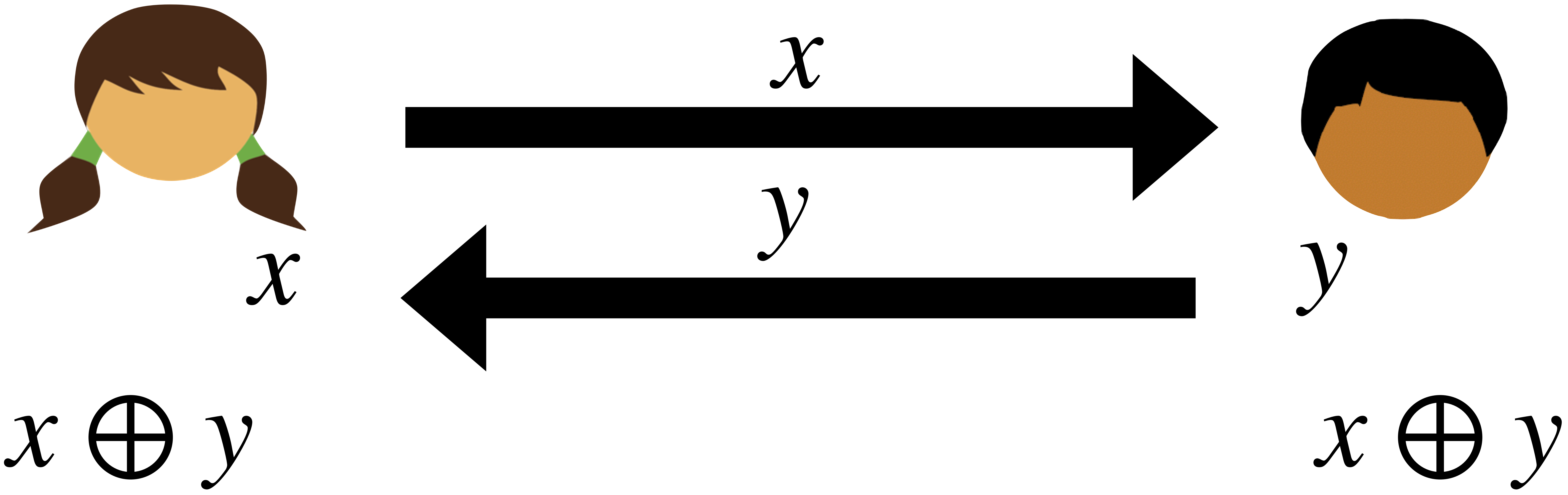
```
secret ←$ {0,1}ⁿ

guess(x : {0,1}ⁿ):
  return x = secret
```

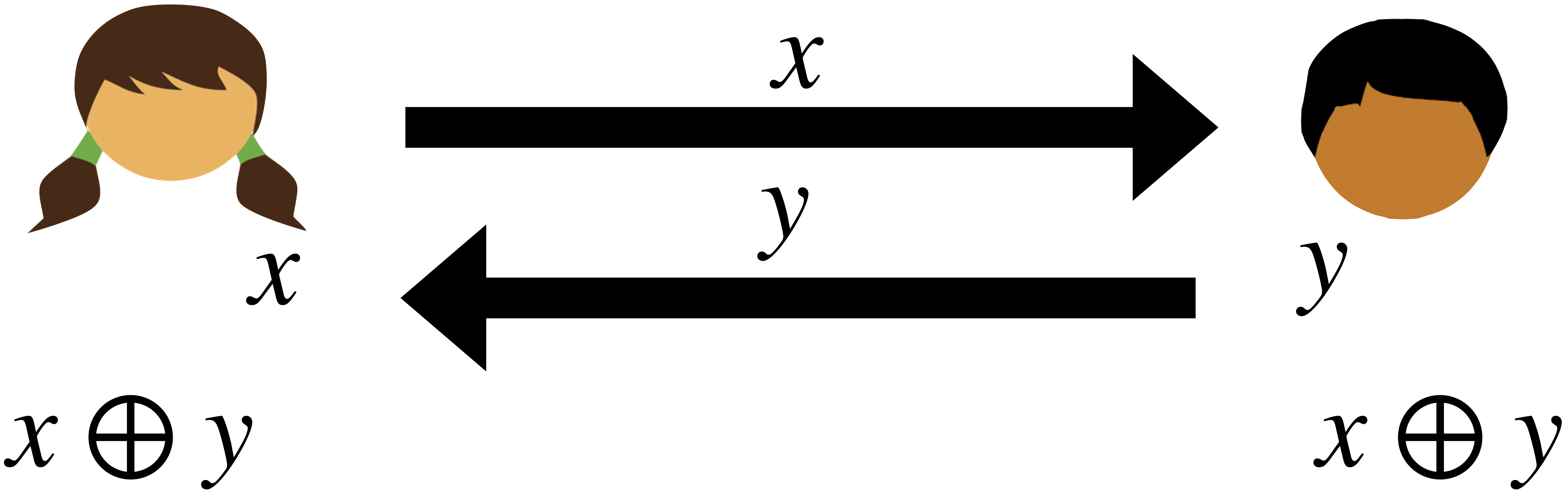
In which sense are these two programs are the same?

Two-Party Semi-Honest Security
for deterministic functionalities

Let f be a function. We say that a protocol Π securely computes f in the presence of a ***semi-honest adversary*** *if for each party $i \in \{0,1\}$ there exists a polynomial time* simulator $\mathcal{S}_i$ *such that for all inputs x_0, x_1:*

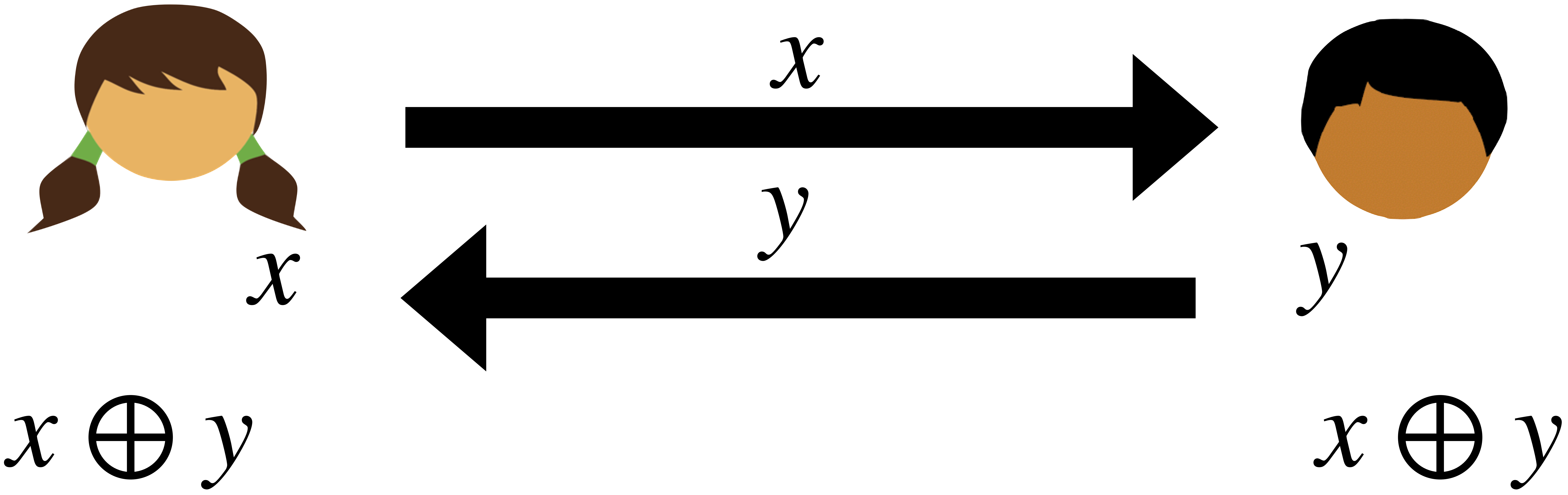
$$\mathrm{View}_i^{\Pi}(x_0, x_1) \stackrel{c}{=} \mathcal{S}_i(x_i, f(x_0, x_1))$$


x
y
x
y
$x \oplus y$
$x \oplus y$



$$\mathrm{View}^{\Pi}_{\mathrm{Bob}}(x, y) = \{x, y\}$$

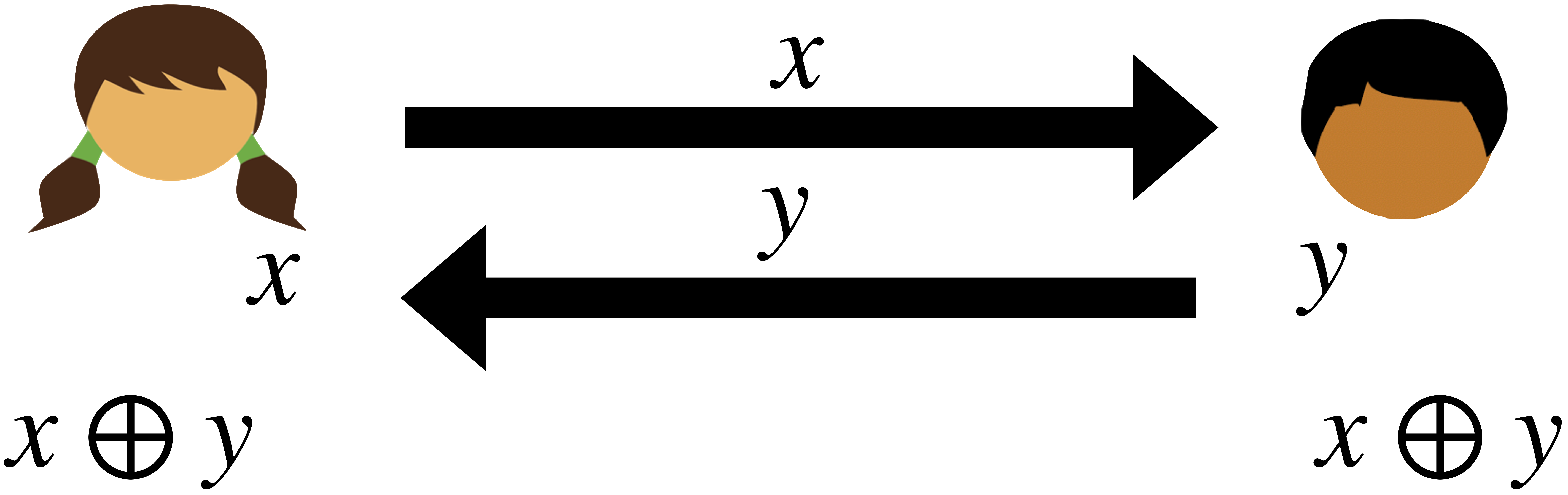
$$\mathrm{Sim}^{\Pi}_{\mathrm{Bob}}(x, x \oplus y) = \{x, (x \oplus y) \oplus y\}$$



$$\mathrm{View}^{\Pi}_{\mathrm{Bob}}(x, y) = \{x, y\}$$

$$\equiv$$

$$\mathrm{Sim}^{\Pi}_{\mathrm{Bob}}(x, x \oplus y) = \{x, (x \oplus y) \oplus y\}$$



$$\mathrm{View}^{\Pi}_{\mathrm{Bob}}(x, y) = \{x, y\}$$

$$\mathrm{Sim}^{\Pi}_{\mathrm{Bob}}(x, x \oplus y) = \{x, z \quad ; \quad z \leftarrow_{\$} \{0,1\}\}$$

Exercise: *Is this a good simulator?*

Lesson:

*Inputs are **not** random. In general, we do not make assumptions about how inputs are distributed*

We should assume the adversary might have side information about the input.

How To Simulate It – A Tutorial on the Simulation Proof Technique*

Yehuda Lindell

Dept. of Computer Science
Bar-Ilan University, ISRAEL
lindell@biu.ac.il

April 25, 2021

Abstract

One of the most fundamental notions of cryptography is that of *simulation*. It stands behind the concepts of semantic security, zero knowledge, and security for multiparty computation. However, writing a simulator and proving security via the use of simulation is a non-trivial task, and one that many newcomers to the field often find difficult. In this tutorial, we provide a guide to how to write simulators and prove security via the simulation paradigm. Although we have tried to make this tutorial as stand-alone as possible, we assume some familiarity with the notions of secure encryption, zero-knowledge, and secure computation.

Keywords: secure computation, the simulation technique, tutorial

*This tutorial appeared in the book *Tutorials on the Foundations of Cryptography*, published in honor of Oded Goldreich's 60th birthday.

Today’s objectives

Review probability distributions/ensembles

Define negligible functions

Introduce indistinguishability

Formalize semi-honest security