CIS 5560

Cryptography
Lecture 3

Course website:

pratyushmishra.com/classes/cis-5560-s24/

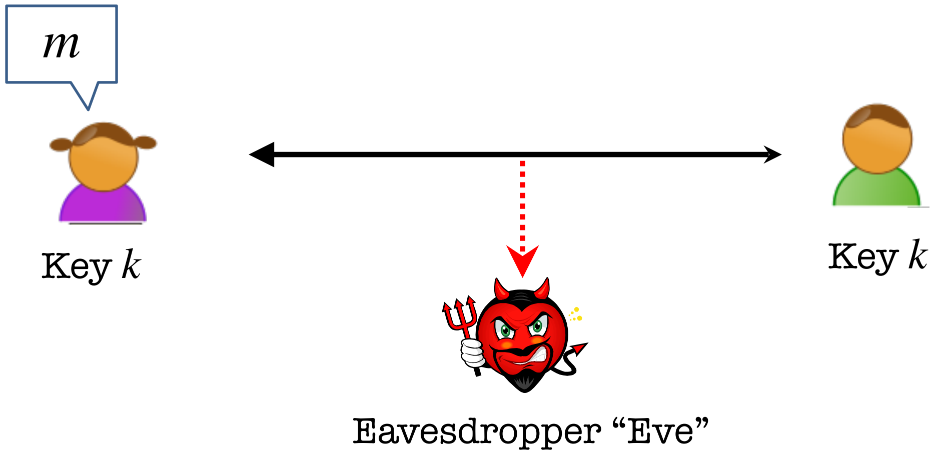
Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Announcements

- **HW 1 is out;** due Monday, Jan 29 at 5PM on Gradescope
  - Covers OTPs and negligible functions (this class)
  - Get started today and make use of office hours!
- Cryptography related CIS Colloquium on Tuesday (1/30) after class
  - See what high level cryptography research looks like!
  - Bonus point on next week's homework if you attend!

Recap of last lecture

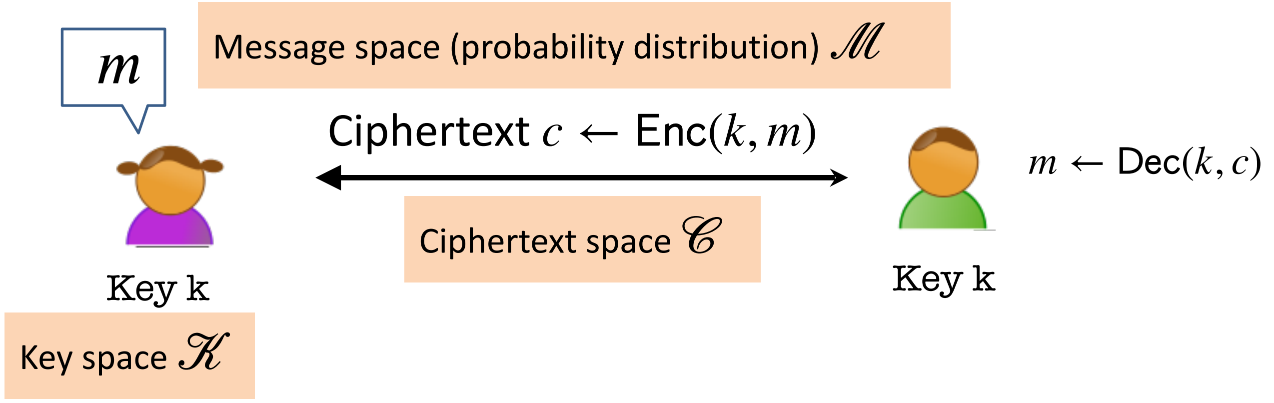
Secure Communication



Alice wants to send a message m to Bob without revealing it to Eve.

Key Notion: Secret-key Encryption

(or Symmetric-key Encryption)



Three (possibly randomized) polynomial-time algorithms:

- **Key Generation Algorithm:** $\mathsf{Gen}(1^k) \rightarrow k$

- **Encryption Algorithm:** $\mathsf{Enc}(k, m) \rightarrow c$

- **Decryption Algorithm:** $\mathsf{Dec}(k, c) \rightarrow m$

The Axiom of ~~Modern Crypto~~ Life

Feasible Computation = randomized polynomial-time* algorithms

(**p.p.t.** = Probabilistic polynomial-time)

(polynomial in a security parameter n)

* in recent years, quantum polynomial-time

Computational Indistinguishability

World 0:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_0)$

World 1:

$k \leftarrow \mathcal{K}$

$c = \mathsf{Enc}(k, m_1)$

Eve is arbitrary **PPT** **distinguisher**.

She needs to decide whether c came from World 0 or World 1.

For every **PPT** Eve, there exists a negligible fn ε, st for all m_0, m_1,

$$\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r} k \leftarrow \mathcal{K} \\ b \leftarrow \{0,1\} \\ c = \mathsf{Enc}(k, m_b) \end{array}\right] < \frac{1}{2} + \varepsilon(n)$$

Called “advantage”

New Notion: Negligible Functions

Functions that grow slower than $1/p(n)$ for any polynomial p.

Definition: A function $\varepsilon : \mathbb{N} \to \mathbb{R}$ is **negligible** if
for every polynomial function p,
there exists an n_0 s.t.
for all $n > n_0$:

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

<u>**Key property:**</u> Events that occur with negligible probability look ***to poly-time algorithms*** like they ***never*** occur.

PRG Def 1: Indistinguishability

Definition [Indistinguishability]:

A deterministic polynomial-time computable function

$G : \{0,1\}^n \to \{0,1\}^m$ is a **PRG** if:

(a) It is expanding: $m > n$ and

(b) for every PPT algorithm D (called a distinguisher) if there is a negligible function ε such that:

$$\left| \Pr[D(G(U_n)) = 1] - \Pr[D(U_m) = 1] \right| = \varepsilon(n)$$

Notation: U_n (resp. U_m) denotes the random distribution on n-bit (resp. m-bit) strings; m is shorthand for $m(n)$.

Today's Lecture

- Semantic security
- PRGs → Semantically-secure encryption
- Constructions of PRGs
    - Real-world schemes
    - Theoretical constructions

Semantic Security

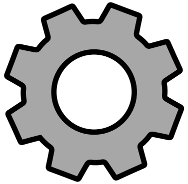
For every **PPT** Eve, there exists a negligible fn ε, st for all m_0, m_1,

$$\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r} k \leftarrow \mathscr{K} \\ b \leftarrow \{0,1\} \\ c := \mathsf{Enc}(k, m_b) \end{array}\right] < \frac{1}{2} + \varepsilon(n)$$

Last time, we briefly discussed that we can view this as a game between a "challenger" and the adversary Eve. Let's flesh that out.

Semantic Security

Eve



c

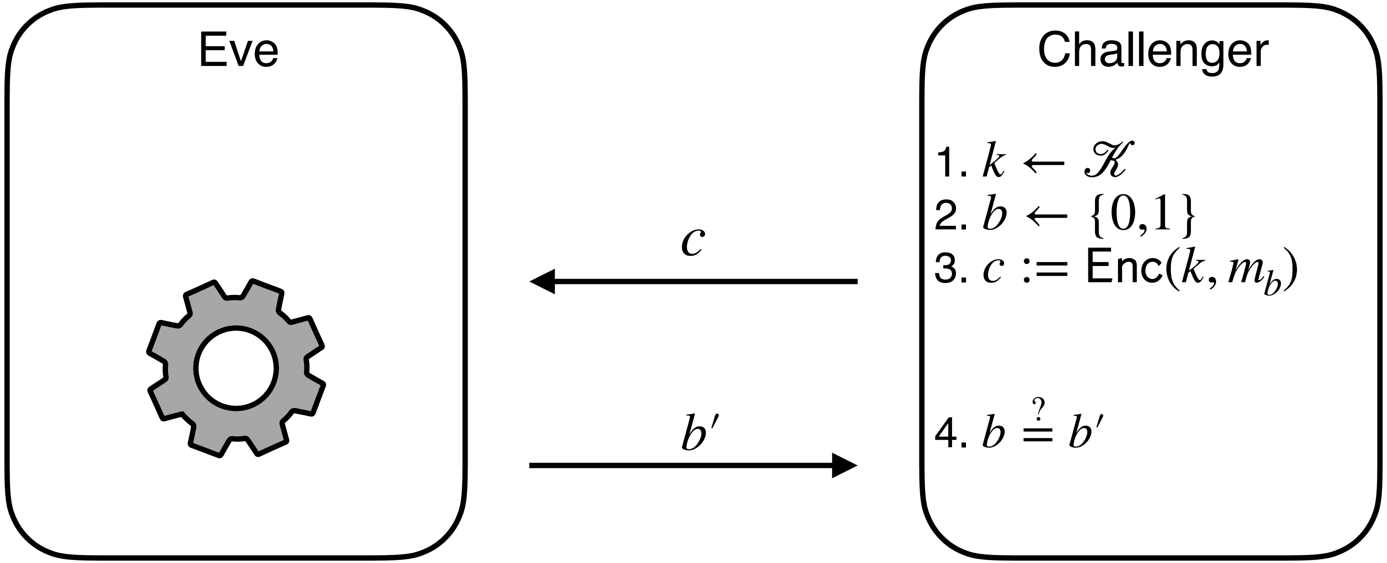
b'

Challenger

1. $k \leftarrow \mathscr{K}$
2. $b \leftarrow \{0,1\}$
3. $c :=$ Enc(k, m_b)

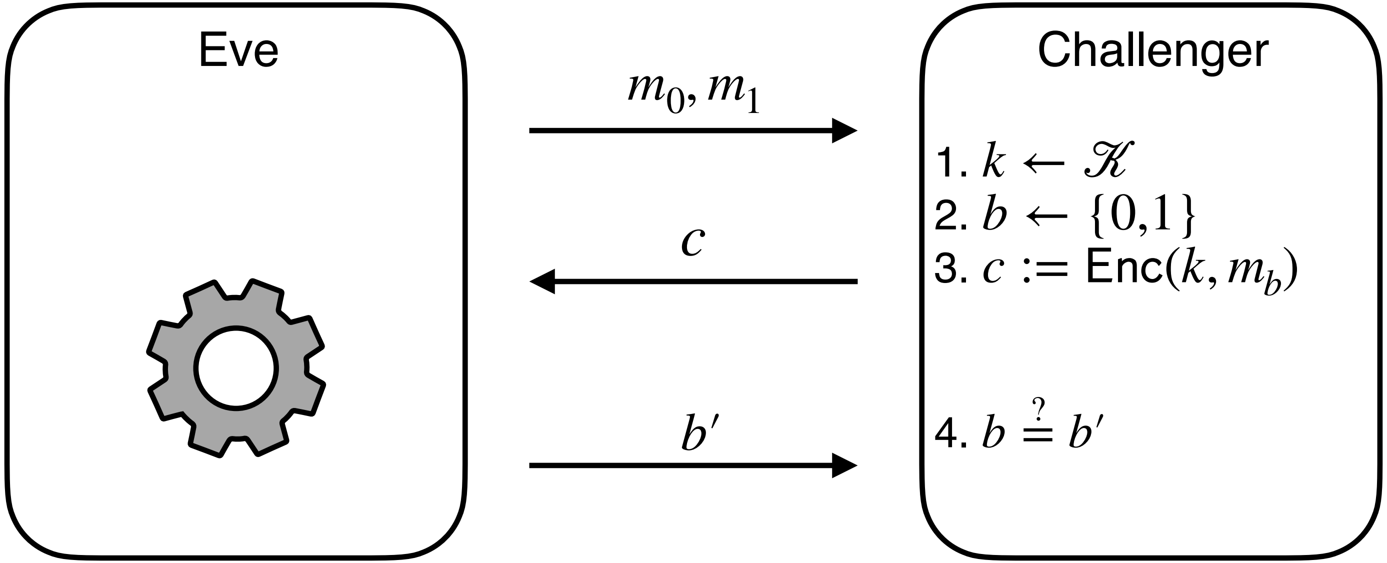
4. $b \stackrel{?}{=} b'$

Semantic Security



We had a good question last time: how does Eve even know what the choices for m_0, m_1 are?

Semantic Security



Ans: we'll let Eve choose the messages!

Semantic Security

For every **PPT** Eve, there exists a negligible fn ε such that

$$\Pr\left[\mathsf{Eve}(c) = b \,\middle|\, \begin{array}{r} (m_0, m_1) \leftarrow Eve \\ k \leftarrow \mathcal{K} \\ b \leftarrow \{0,1\} \\ c := \mathsf{Enc}(k, m_b) \end{array} \right] < \frac{1}{2} + \varepsilon(n)$$

Semantic Security

For every **PPT** Eve, there exists a negligible fn ε such that

$$\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r} (m_0, m_1) \leftarrow Eve \\ k \leftarrow \mathscr{K} \\ b \leftarrow \{0,1\} \\ c := \mathsf{Enc}(k, m_b) \end{array}\right] < \frac{1}{2} + \varepsilon(n)$$

Why is this the “right” definition?

Intuitively: even if Eve knows which messages are candidate plaintexts, ciphertext *still* reveals no information!

PRGs → Semantically Secure Encryption

PRG $\implies$ Semantically Secure Encryption

(or, How to Encrypt n+1 bits using an n-bit key)

- Gen$(1^k) \to k$:
  - Sample an n-bit string at random.
- Enc$(k, m) \to c$:
  - Expand k to an $n+1$-bit string using PRG: $s = G(k)$
  - Output $c = s \oplus m$
- Dec$(k, c) \to m$:
  - Expand k to an $n+1$-bit string using PRG: $s = G(k)$
  - Output $m = s \oplus c$

Correctness:

$Dec(k, c)$ outputs $G(k) \oplus c = G(k) \oplus G(k) \oplus m = m$

PRG $\Longrightarrow$ Semantically Secure Encryption

Security: your first reduction!

Suppose for contradiction that there exists an Eve that breaks our scheme.

That, is assume that there is a p.p.t. Eve, and polynomial function p s.t.

$$\Pr\left[\mathsf{Eve}(c) = b \;\middle|\; \begin{array}{r} (m_0, m_1) \leftarrow \mathsf{Eve} \\ k \leftarrow \mathscr{K} \\ b \leftarrow \{0,1\} \\ c := \mathsf{Enc}(k, m_b) \end{array}\right] > \frac{1}{2} + 1/p(n)$$

PRG $\Longrightarrow$ Semantically Secure Encryption

Security: your first reduction!

Assume that there is a p.p.t. Eve, a polynomial function p and m_0, m_1 s.t.

$$\Pr\left[\text{Eve}(c) = b \;\middle|\; \begin{array}{r} (m_0, m_1) \leftarrow \text{Eve} \\ k \leftarrow \{0,1\}^n \\ b \leftarrow \{0,1\} \\ c := G(k) \oplus m_b \end{array}\right] > \frac{1}{2} + 1/p(n)$$

Let's call this ρ

$$\text{Compare with } \Pr\left[\text{Eve}(c) = b \;\middle|\; \begin{array}{r} (m_0, m_1) \leftarrow \text{Eve} \\ k' \leftarrow \{0,1\}^{n+1} \\ b \leftarrow \{0,1\} \\ c := k' \oplus m_b \end{array}\right] = \frac{1}{2}$$

Let's call this ρ'

Clearly, Eve can break security in
PRG case, but not in OTP world!

$$\downarrow$$

Eve can distinguish pseudorandom from random!

Idea: Use Eve to break PRG indistinguishability!

Distinguisher $D(y)$:

1. Sample two messages m_0, m_1, and a bit b
2. Compute $b' \leftarrow \text{Eve}(y \oplus m_b)$
3. If $b' = b$, output "PRG"
4. Otherwise, output "Random"

World 0

$\Pr[D \text{ outputs "PRG"} \mid y \text{ is pseudorandom}]$
$= \Pr[\text{Eve outputs } b' = b \mid y \text{ is pseudorandom}]$
$= \rho \geq 1/2 + 1/p(n)$

World 1

$\Pr[D \text{ outputs "PRG"} \mid y \text{ is random}]$
$= \Pr[\text{Eve outputs } b' = b \mid y \text{ is random}]$
$= \rho' = 1/2$

Therefore,

$$\left| \Pr[D \text{ outputs "PRG"} \mid y \text{ is pseudorandom}] - \Pr[D \text{ outputs "PRG"} \mid y \text{ is random}] \right|$$
$$\geq 1/p(n) \qquad \blacksquare$$

PRG $\Longrightarrow$ Semantically Secure Encryption

(or, How to Encrypt n+1 bits using an n-bit key)

***Q1*:** Do PRGs exist?

(Exercise: If P=NP, PRGs do not exist.)

***Q2*:** How do we encrypt longer messages or many messages with a fixed key?

(**Length extension**: If there is a PRG that stretches by one bit, there is one that stretches by polynomially many bits)

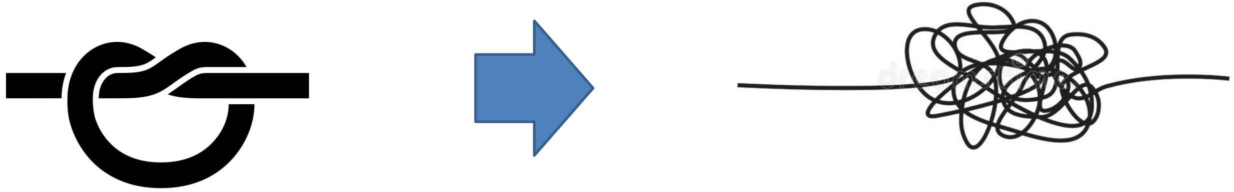
(**Pseudorandom functions**: PRGs with exponentially large stretch and "random access" to the output.)

$\boldsymbol{Q1}$: Do PRGs exist?

Constructing PRGs: Two Methodologies

<u>The Practical Methodology</u>

1. Start from a design framework
(e.g. “appropriately chosen functions composed appropriately many times look random”)

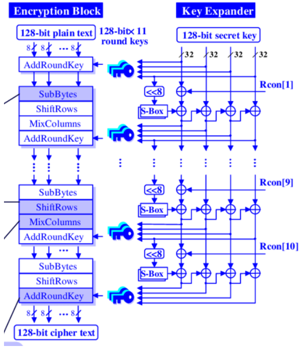


Constructing PRGs: Two Methodologies

<u>**The Practical Methodology**</u>

1. Start from a design framework
(e.g. “appropriately chosen functions composed appropriately many times look random”)

2. Come up with a candidate construction



Rijndael
(now the Advanced Encryption Standard)

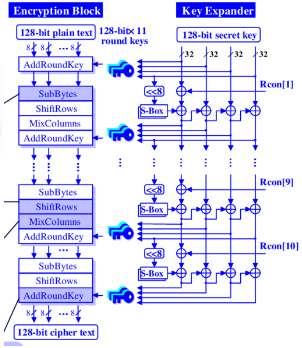
Constructing PRGs: Two Methodologies

<u>The Practical Methodology</u>

1. Start from a design framework
(e.g. “appropriately chosen functions composed appropriately many times look random”)

2. Come up with a candidate construction

3. Do extensive cryptanalysis.



Examples

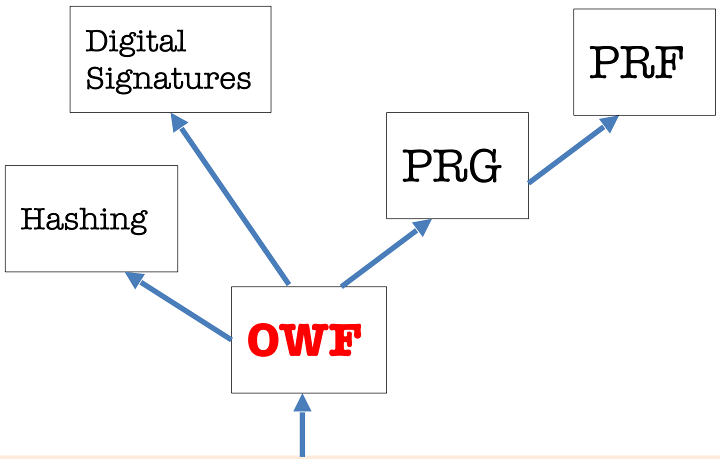
- **RC4: old PRG from 1987**
  - Proposed by Ron Rivest (of RSA fame)
  - **Fast and simple**
  - Used in TLS till 2013
  - However lots of biases
    - e.g. 2nd byte of output has 2/256 chance of being 0.
  - In 2013, attack which made key recovery feasible with just 2^{20} ciphertexts!
  - Finally deprecated in 2015, *28 years* after creation!

Constructing PRGs: Two Methodologies

<u>**The Foundational Methodology (much of this course)**</u>

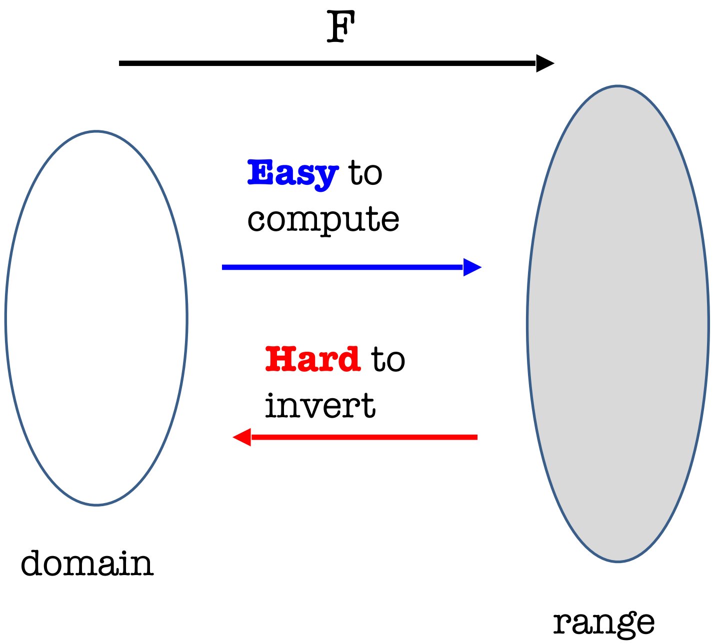
Reduce to simpler primitives.

“Science wins either way” –Silvio Micali



well-studied, average-case hard, problems

One-way Functions (Informally)



Source of all hard problems in cryptography!

What is a good definition?

One-way Functions (Take 1)

A function (family) $\{F_n\}_{n\in\mathbb{N}}$ where $F(\cdot):\{0,1\}^n \to \{0,1\}^{m(n)}$ is **one-way** if for every p.p.t. adversary A, the following holds:

$$\Pr\left[A(1^n, y) = x \,\middle|\, \begin{array}{c} x \leftarrow \{0,1\}^n \\ y := F_n(x) \end{array}\right] = \mathsf{negl}(n)$$

Consider $F_n(x) = 0$ for all x.

This is one-way according to the above definition.
In fact, impossible to find *the* inverse even if A has unbounded time.

Conclusion: not a useful/meaningful definition.

One-way Functions (Take 1)

A function (family) $\{F_n\}_{n\in\mathbb{N}}$ where $F(\cdot) : \{0,1\}^n \to \{0,1\}^{m(n)}$ is **one-way** if for every p.p.t. adversary A, the following holds:

$$\Pr\left[A(1^n, y) = x \;\middle|\; \begin{array}{r} x \leftarrow \{0,1\}^n \\ y := F_n(x) \end{array}\right] = \mathrm{negl}(n)$$

The Right Definition: Impossible to find ***an*** inverse efficiently.

One-way Functions: The Definition

A function (family) $\{F_n\}_{n\in\mathbb{N}}$ where $F(\cdot) : \{0,1\}^n \to \{0,1\}^{m(n)}$ is **one-way** if for every p.p.t. adversary A, the following holds:

$$\Pr\left[F_n(x') = y \;\middle|\; \begin{array}{r} x \leftarrow \{0,1\}^n \\ y := F_n(x) \\ x' \leftarrow A(1^n, y) \end{array}\right] = \mathsf{negl}(n)$$

- Can always find *an* inverse with unbounded time
- … but should be hard with probabilistic polynomial time

One-way Permutations:

One-to-one one-way functions with $m(n) = n$.

How to get PRG from OWF?

OWF → PRG, Attempt #1

PRG(k)

1. Output $F_n(k)$

(Assume $m(n) > n$)

Does this work?

OWF → PRG, Attempt #1

Consider $F_n(x)$ constructed from another OWF F'_n:

1. Compute $y := F'_n(x)$
2. Output $y' := (y_0,\ 1, y_1,\ 1, \ldots, y_n,\ 1)$

PRG(k)

1. Output $F_n(k)$

Is F one-way?

Yes!

Is PRG **unpredictable?**

No!

Our problem:

OWFs don’t tell us anything about how their inputs are distributed

They are only hard to invert

Next class

- How to get randomness from OWF output
  - How to use this to get PRGs
- How to extend the length of PRGs
- How to get PRGs with “exponentially-large” output