CIS 5560

Cryptography
Lecture 6

Announcements

- **HW 2 out tomorrow**
  - Due **Friday**, Feb 13 at 5PM on Gradescope
  - Covers PRGs, PRFs, multi-message indistinguishability
- HW1 due this Friday (Feb 6)

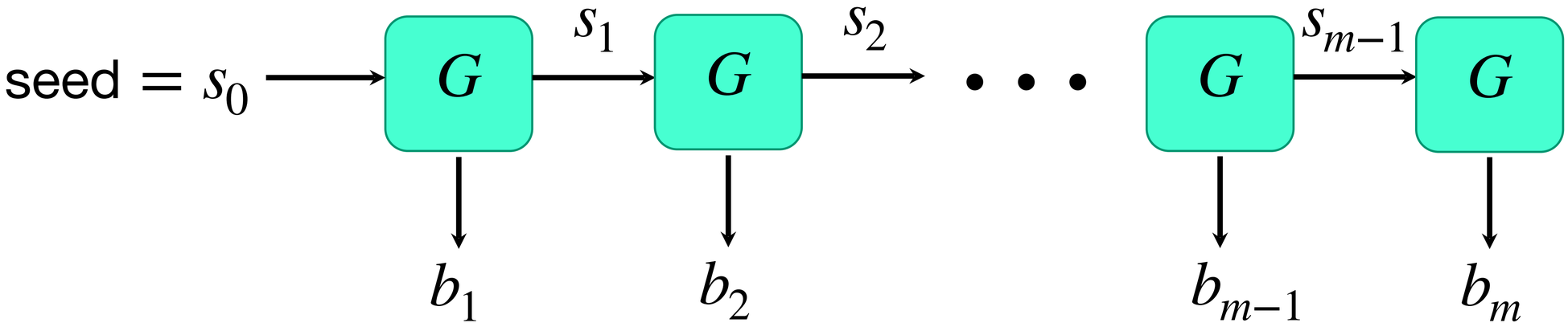
Recap of last lecture

Construction: PRG Length extension

Let $G : \{0,1\}^n \to \{0,1\}^{n+1}$ be a PRG

Goal: use G to generate **many** pseudorandom bits.

Construction of $G'(s_0)$:



Technique: Hybrid argument

Key idea: instead of directly trying to go from first distribution to second, take small steps!

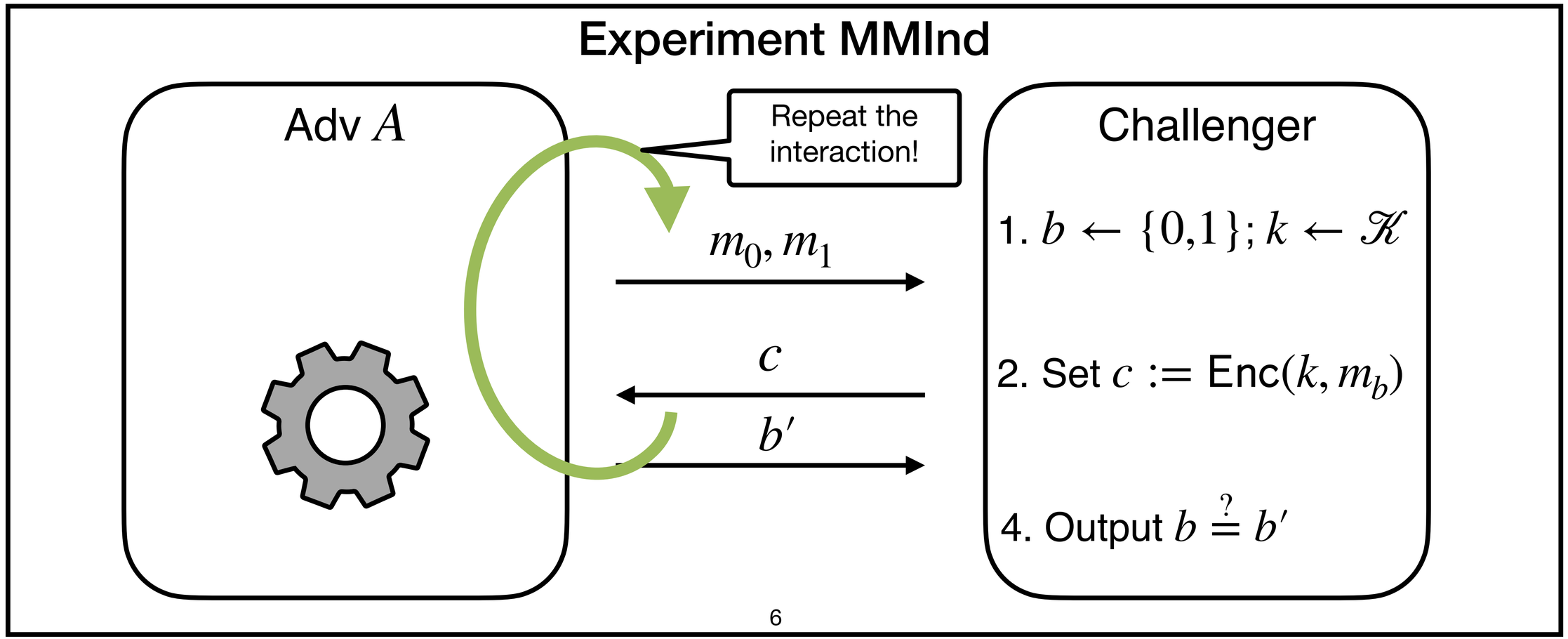
1. **Construct the steps:**
   A sequence of (polynomially-many) distributions $H_1, \ldots, H_{m-1}$ b/w the two target distributions.
2. **Show that it's easy to move between steps:**
   Argue that each pair of neighboring distributions are indistinguishable.
3. **Start moving:**
   Conclude that the target distributions are indistinguishable via contradiction:
   A. Assume the target distributions are distinguishable
   B. Must be the case that an intermediate pair of distributions is distinguishable
   C. This contradicts 2 above.

New: Multi-msg Indistinguishability Game

For every PPT "distinguishing" adversary A

$$|\Pr[\text{MMInd} = 1] - \Pr[\text{random guess}]| = \text{negl}(\lambda)$$

"Advantage"



New: Multi-msg Indistinguishability Game

For every **PPT** A, there exists a negligible fn ε,

$$\left| \Pr\left[A(c_q) = b \;\middle|\; \begin{array}{r} k \leftarrow \mathscr{K}, b \leftarrow \{0,1\} \\ \text{For } i \text{ in } 1,\ldots,q: \\ (m_{i,0}, m_{i,1}) \leftarrow A(c_{i-1}) \\ c_i = \mathsf{Enc}(k, m_{i,b}) \end{array} \right] - \frac{1}{2} \right| < \varepsilon(n)$$

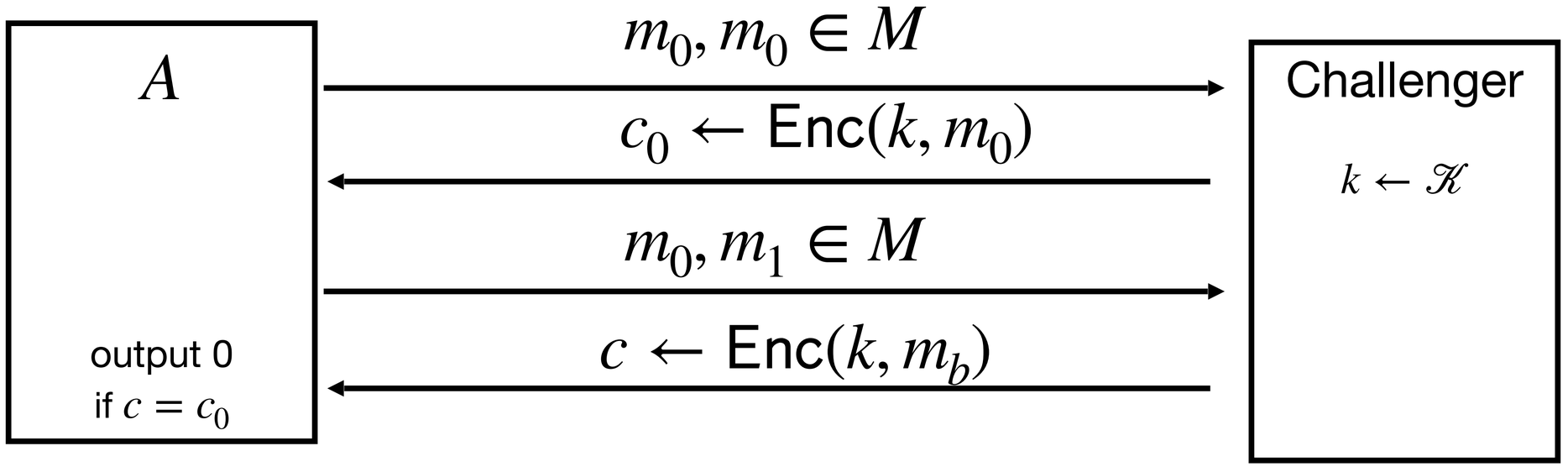
Indistinguishability under “Chosen-Plaintext Attack” IND-CPA

Today's Lecture

- Encryption for many messages
  - Definition
  - Attempted construction from PRGs
- PRFs
- PRPs
- Block ciphers

Stream Ciphers (PseudoOTP) insecure under CPA

Problem: Enc(k, m) outputs same ciphertext for msg m.



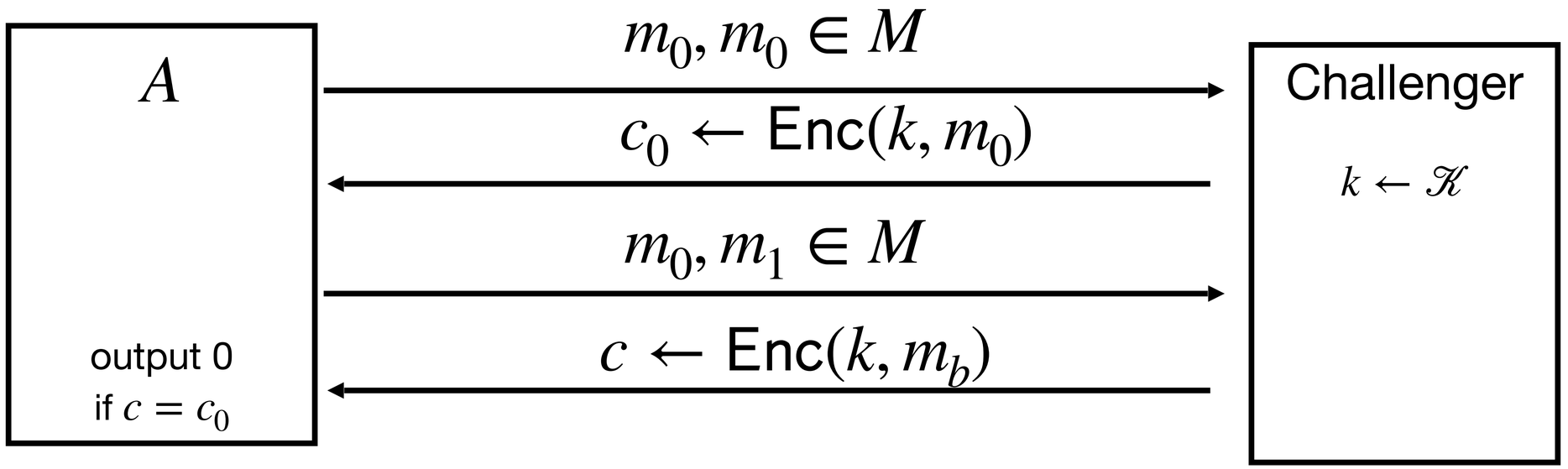
So what?

an attacker can learn that two encrypted files are the same, two encrypted packets are the same, etc.

Leads to significant attacks when message space is small

Stream Ciphers insecure under CPA

Problem: Enc(k, m) outputs same ciphertext for msg m.



If secret key is to be used multiple times

given the same plaintext message twice, encryption must produce different outputs.

Ideas for multi-message encryption

How to make encryption of same messages change?

Problem: If encrypting the same message twice, all inputs are the same: key, message.

Solution: we need to add new inputs that change per encryption! What can we change?

- State? (e.g. counter of num msgs)
- Randomness?

Approach 1: Stateful encryption

$\text{Gen}(1^\lambda) \to k$:

1. Sample an n-bit string at random.

$\text{Enc}(k, m, \mathbf{st}) \to c$:

1. Expand k to an $n+1$-bit string using PRG: $s = G(k)$
2. Discard first ℓ bits of s to get s'
3. Set $\ell := \ell + 1$
4. Output $c = s' \oplus m$

$\text{Dec}(k, c) \to m$:

1. Repeat steps 1—4 of Enc
2. Output $m = s' \oplus c$

Is this secure for multiple messages?

Does this work?

Ans: Yes!

Exercise: reduce to PRG security

Idea: Encryption of i-th message is PseudoOTP.

Pros:

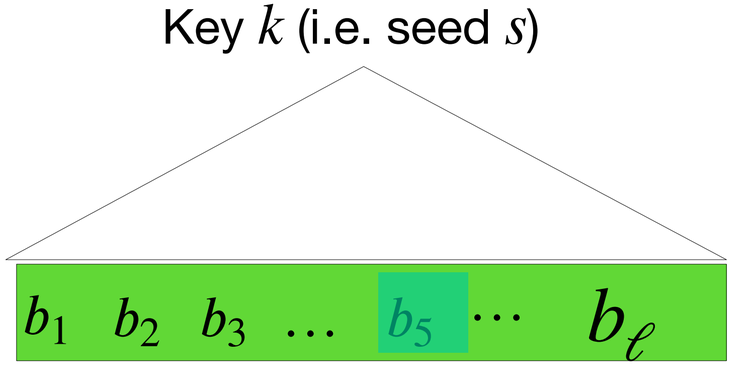
- Relies on existing tools
- Generally fast

Cons:

- Must maintain counter of encrypted messages
- Must rerun PRG from start every time
- Sequential encryption/decryption

Problem: PRGs are sequential

PRG $G(k)$



- With a PRG, accessing the ℓ-th bit takes time ℓ.
- How to get efficient *random access* into output?
- That is, we want some function such that $F(\ell) = \ell$-th bit

New tool:

Pseudorandom Function

Background: Random function

- Let X be an input space, and Y be an output space.
- We will denote the set of all functions from X to Y as $\mathsf{Fns}[X, Y]$
  - The number of such functions is $|Y|^{|X|}$.
- A random function from X to Y is a function that is sampled uniformly at random from $\mathsf{Fns}[X, Y]$
- Important property of every random function f:
  - For each $x \in X$, $f(x)$ is uniformly and independently distributed in Y.

Stateful encryption w/ RFs

Gen$(1^n) \to k$: Sample a random function f and set $k := f$.

Enc$(k, m, \mathbf{st}) \to c$:

1. Interpret $\mathbf{st}$ as number ℓ of messages encrypted so far.
2. Output $c = f(\ell) \oplus m$

Dec$(k, c, \mathbf{st}) \to m$:

1. Interpret $\mathbf{st}$ as number ℓ of messages encrypted so far.
2. Output $m = f(\ell) \oplus c$

Does this work?

Ans: Yes!

Idea: Each encryption is XOR-ing with output of RF;
i.e., XOR-ing with a uniformly random string

Pros:

- Relies on existing tools
- Generally fast
- No need to run RF from start!

Cons:

- Must maintain counter of encrypted messages
- **How to store a random function?**

Encryption w/ RFs

- What's the problem with this?
- Hint: What does a random function look like?
  - Is it efficiently evaluatable?
  - Does it have a short description?

Problem: Random Functions can't be stored efficiently

A random function is a random mapping from X to Y.

Simplest representation: function table

What is the size of an arbitrary mapping?

$$|X| \log |Y|$$

For each x, $|Y|$ possible choices;
each choice has $\log |Y|$ bits representation

Problem: Random Functions can't be stored efficiently

For encryption, $|X|\log|Y|$ is too large!

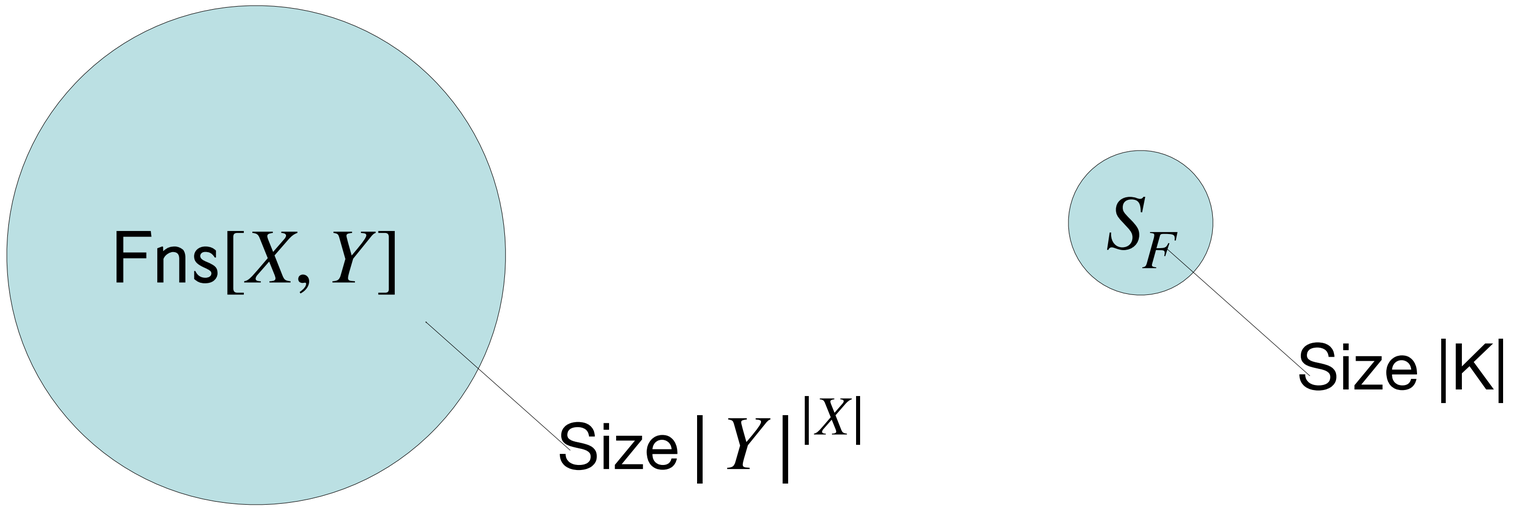
Let's see why:

In our case, $|Y|$ is message length, e.g. $Y = \{0,1\}$, $|Y| = 2$.
if we encrypt, e.g., $|X| = 2^{20}$ 1-bit messages, our key is now 2^{20} bits, i.e. same as OTP!

Also, $|Y|^{|X|}$ should be large (otherwise brute force possible: try all possible functions).

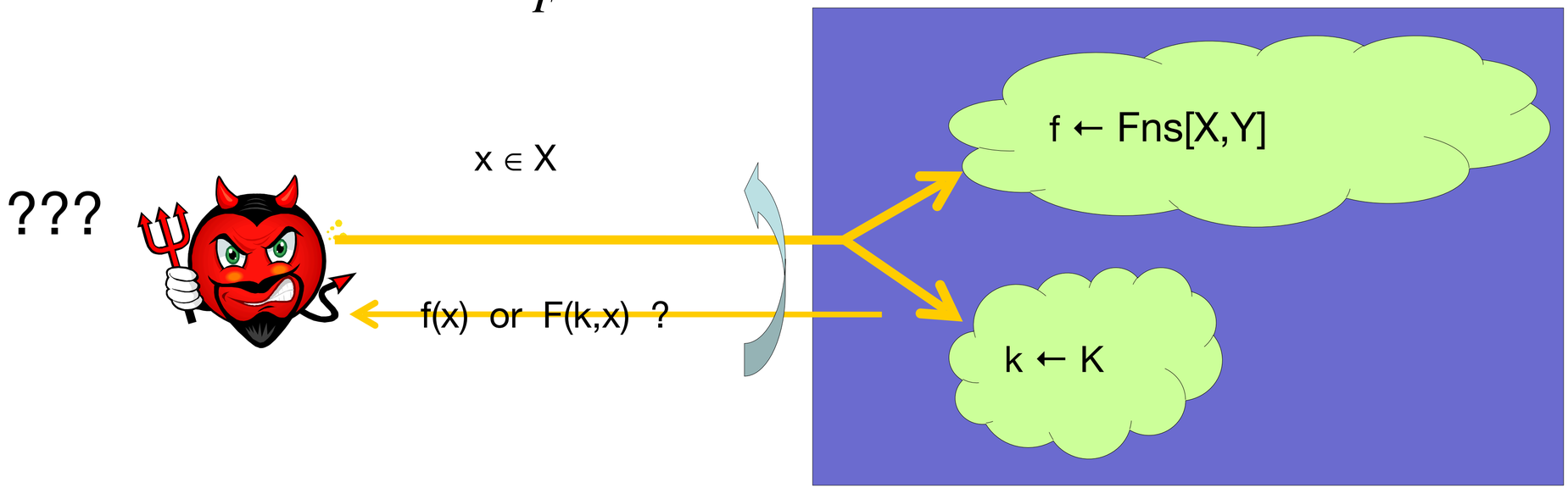
Solution: *Pseudo*random functions

- Replace a real random function with a function that *looks* random
- $S_F = \{F(k, \cdot) \mid k \in \mathcal{K}\} \subset \mathsf{Fns}[X, Y]$
- Intuition: a PRF is **secure** if
  a random function in $\mathsf{Fns}[X, Y]$ is indistinguishable from
  a random function in S_F



Secure PRFs

- Replace a real random function with a function that *looks* random
- $S_F = \{F(k, \cdot) \mid k \in \mathcal{K}\} \subset \mathrm{Fns}[X, Y]$
- Intuition: a PRF is **secure** if
  a random function in $\mathrm{Fns}[X, Y]$ is indistinguishable from
  a random function in S_F

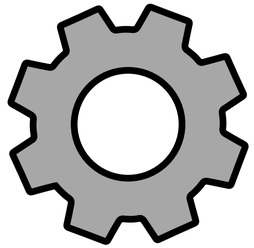


How to define PRF security?

- For PRG security, we give the adversary either a random string or a pseudorandom string, and ask it to figure out which one it is
- Can the same strategy work for PRFs?

PRF Security - Attempt 1

Adv $\mathcal{A}$



f

b'

Challenger

1. $b \leftarrow \{0,1\}$
2. If b = 0
   1. Sample $f \leftarrow \text{Fns}[X, Y]$
3. If b = 1
   1. Sample $k \leftarrow \mathcal{K}$
   2. Set $f(\cdot) = F(k, \cdot)$

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

$$|\Pr[b = b'] - 1/2| = \text{negl}(n)$$

Problem: Random Functions can't be stored efficiently

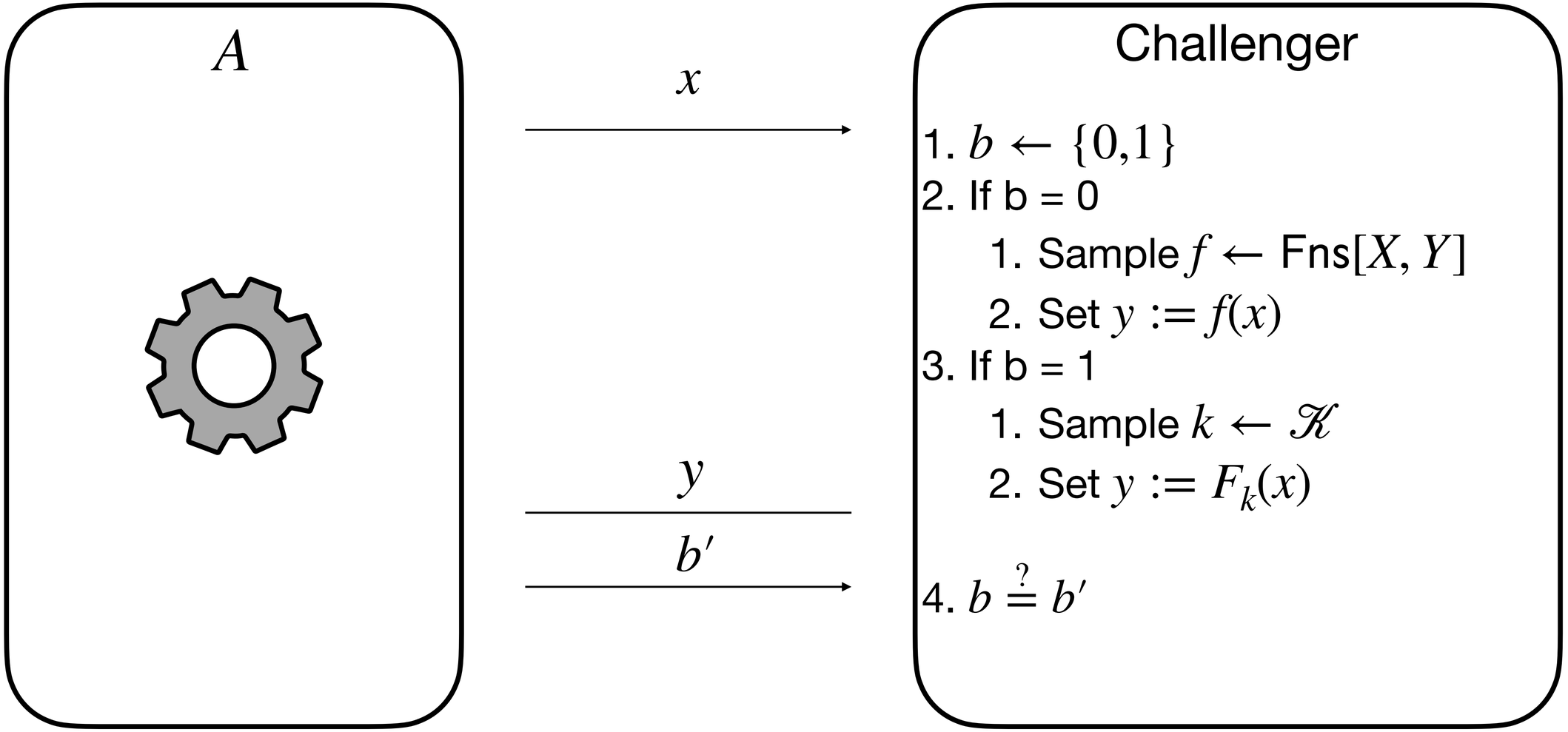
For encryption, $|X|\log|Y|$ is too large!

Let's see why:

In our case, $|Y|$ is message length, e.g. $Y = \{0,1\}$, $|Y| = 2$.
if we encrypt, e.g., $|X| = 2^{20}$ 1-bit messages, our key is now 2^{20} bits, i.e. same as OTP!

Also, $|Y|^{|X|}$ should be large (otherwise brute force possible: try all possible functions).

PRF Security



$$|\Pr[b = b'] - 1/2| = \mathsf{negl}(n)$$

PRF Security - Attempt 2

- Q: How many questions should the adversary be allowed to ask?
  - 1
  - 2
  - poly(n)
  - exp(n)

- Why is 1 insufficient? Can't tell any information from 1 query
- Why is exp(n) too many? Adv will run in exponential time!

PRF Security Game

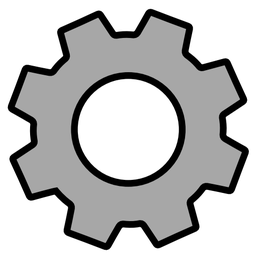
For every PPT "distinguishing" adversary A

$$|\Pr[\text{PRFInd} = 1] - \Pr[\text{random guess}]| = \text{negl}(\lambda)$$

"Advantage"

Experiment PRFInd

Adv A



Repeat the interaction!

x →

← y

b' →

Challenger

1. $b \leftarrow \{0,1\}$
2. If b = 0
   1. Sample $f \leftarrow \text{Fns}[X, Y]$
   2. Set $y := f(x)$
3. If b = 1
   1. Sample $k \leftarrow \mathcal{K}$
   2. Set $y := F_k(x)$
4. $b \stackrel{?}{=} b'$

Equivalent definition:

For every PPT "distinguishing" adversary A

$$|\Pr[A^{F_k(\cdot)} \mid k \leftarrow \{0,1\}^n] - \Pr[A^{f(\cdot)} \mid f \leftarrow \mathsf{Funs}[X, Y]]| = \mathsf{negl}(\lambda)$$

Oracle or "opaque" access

"Advantage"

PRFs → multi-message encryption

Ideas for multi-message encryption

- State? (e.g. counter of num msgs)
- Randomness?

Stateful encryption w/ PRFs

Gen$(1^n) \to k$:

Sample an n-bit string at random.

Enc$(k, m, \mathbf{st}) \to c$:

1. Interpret $\mathbf{st}$ as number ℓ of messages encrypted so far.
2. Output $c = F_k(\ell) \oplus m$

Dec$(k, c, \mathbf{st}) \to m$:

1. Interpret $\mathbf{st}$ as number ℓ of messages encrypted so far.
2. Output $m = F_k(\ell) \oplus c$

Does this work?

Ans: Yes!

Idea: Using PRF for encryption is indistinguishable from using RF;

So hybrid H0 is $\mathsf{Enc}(k, m_0)$;

H1 is $\mathsf{Enc}_{\mathsf{RF}}(k, m_0)$;

H2 is $\mathsf{Enc}_{\mathsf{RF}}(k, m_0)$;

H3 is $\mathsf{Enc}(k, m_1)$

Pros:

- Generally fast
- No need to run PRF from start!

Cons:

- Must maintain counter of encrypted messages
  - (Just like PRG solution)

Ideas for multi-message encryption

- State? (e.g. counter of num msgs)
- Randomness?

Randomized encryption w/ PRFs

Gen(1^n): Generate a random n-bit key k that defines $F_k : \{0,1\}^{\ell} \to \{0,1\}^m$

Enc(k, m): Pick a random r and
set the ciphertext $c := (r, y = F_k(r) \oplus m)$

Dec$(k, c = (r, y))$: Output $F_k(r) \oplus c$

Does this work?

Ans: Yes!

Proof: next

Pros:

- Relies on existing tools
- Generally fast
- No need to run PRF from start!

Cons:

- Need good randomness during encryption

Security of Randomized Encryption

Enc(k, m): Pick a random r and set the ciphertext $c := (r, y = F_k(r) \oplus m)$

Dec$(k, c = (r, y))$: Output $F_k(r) \oplus c$

- **Proof strategy:** Focusing on 1msg security first

Proof by hybrid argument

Enc(k, m): Pick a random r and set the ciphertext $c := (r, y = F_k(r) \oplus m)$

Dec$(k, c = (r, y))$: Output $F_k(r) \oplus c$

Single msg security says that the following dists are indistinguishable.

$\{c \leftarrow \text{Enc}(k, m_0) \mid k \leftarrow \mathcal{K}\}$ and $\{c \leftarrow \text{Enc}(k, m_1) \mid k \leftarrow \mathcal{K}\}$

How to do this? Let's create more (supposedly) indistinguishable distributions:

$H_0 = \{c := (r, m_0 \oplus F_k(r) \mid r \leftarrow \{0,1\}^n; k \leftarrow \mathcal{K}\}$

$H_1 = \{c := (r, m_0 \oplus R(r) \mid r \leftarrow \{0,1\}^n; R \leftarrow \text{Fns}\}$ $\approx$ by PRF security

$H_2 = \{c := (r, m_0 \oplus r' \mid r \leftarrow \{0,1\}^n; r' \leftarrow \{0,1\}^n\}$ $\approx$ defn of random fn

$H_3 = \{c := (r, m_1 \oplus r' \mid r \leftarrow \{0,1\}^n; r' \leftarrow \{0,1\}^n\}$ $\approx$ one time pad

$H_4 = \{c := (r, m_1 \oplus R(r) \mid r \leftarrow \{0,1\}^n; R \leftarrow \text{Fns}\}$ $\approx$ defn of random fn

$H_5 = \{c := (r, m_1 \oplus F_k(r) \mid r \leftarrow \{0,1\}^n; k \leftarrow \mathcal{K}\}$ $\approx$ by PRF security

Security of Randomized Encryption

Enc(k, m): Pick a random r and set the ciphertext $c := (r, y = F_k(r) \oplus m)$

Dec$(k, c = (r, y))$: Output $F_k(r) \oplus c$

- **Proof strategy:**
  - 1msg security done.
  - What about multi-msg security?
  - Similar idea:
    - Switch PRF to RF
    - Argue that each encryption is OTP with fresh key (output of RF), and switch from encryptions of $m_{i,0}$ to encryptions of $m_{i,1}$.
  - Caveat: what if we sample the same r for two different ciphertexts?
    - No longer OTP.
    - Thankfully, if input space of PRF is large, this happens with negligible prob.

So far

Multi-msg security via randomized encryption

Pros:

- Relies on existing tools
- Generally fast
- No need to run PRF from start!

Cons:

- Ciphertext is ~2x larger: $(r, m \oplus F_k(r))$
- Can only encrypt fixed-size n bit msg at a time
- Thus, sending a message of, say, $10n$ bits, requires $20n$-sized ciphertext