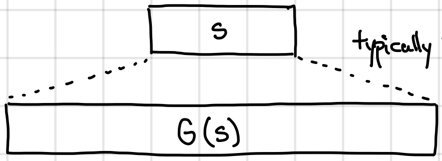
Idea: "compress" the one-time pad: we will generate a long random-looking string from a short seed (e.g., $s \in \{0,1\}^{128}$).



typically: $s \in \{0,1\}^\lambda$ (λ is the seed length or security parameter)

$G(s) \in \{0,1\}^n$ where $n \gg \lambda$

n is the "stretch" of a PRG

Stream cipher: $\mathcal{K} = \{0,1\}^\lambda$

$\mathcal{M} = \mathcal{C} = \{0,1\}^n$

Encrypt (k, m): $c \leftarrow m \oplus \boxed{G(k)}$

Decrypt (k, c): $m \leftarrow c \oplus \boxed{G(k)}$

Instead of xor-ing with the key, we use the key to derive a "stream" of random-looking bits and use that in place of the one-time pad

If $\lambda < n$, then this scheme cannot be perfectly secure! So we need a different notion of security

Intuitively: Want a stream cipher to function "like" a one-time pad to any "reasonable" adversary.

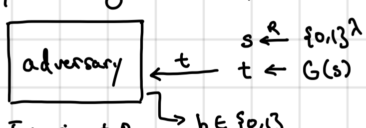
$\Longrightarrow$ Equivalently: output of a PRG should "look" like uniformly-random string

What is a "reasonable" adversary?

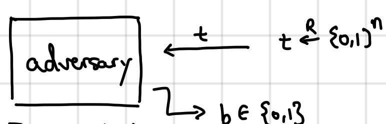
- Theoretical answer: algorithm runs in (probabilistic) polynomial time
- Practical answer: runs in time $< 2^{80}$ and space $< 2^{64}$ (can use larger numbers as well)

Goal: Construct a PRG so no efficient adversary can distinguish output from random.

Captured by defining two experiments or games:



Experiment 0



Experiment 1

the input to the adversary (t) is often called the challenge

Adversary's goal is to distinguish between Experiment 0 (pseudorandom string) and Experiment 1 (truly random string)

↳ It is given as input a string t of length n (either $t \leftarrow G(s)$ or $t \xleftarrow{R} \{0,1\}^n$)

↳ It outputs a guess (a single bit $b \in \{0,1\}$)

Remember: adversary knows the algorithm G; only seed is hidden!

Let $W_0 :=$ Pr[adversary outputs 1 in Experiment 0]

$W_1 :=$ Pr[adversary outputs 1 in Experiment 1]

} define the distinguishing advantage of A as PRGAdv[A, G] $:= |W_0 - W_1|$

DO NOT RELY ON SECURITY BY OBSCURITY!

probabilistic polynomial time

Definition. A PRG $G: \{0,1\}^\lambda \to \{0,1\}^n$ is secure if for all efficient adversaries A,

PRGAdv[A, G] = negl(λ)

↳ negligible function (in the input length)

smaller than any inverse polynomial

e.g., $\frac{1}{2^\lambda}$, $\lambda^{\log \lambda}$

- Theoretical definition: $f(\lambda)$ is negligible if $f \in o(\lambda^c)$ for all $c \in \mathbb{N}$
- Practical definition: quantity $\le 2^{-80}$ or $\le 2^{-128}$

Understanding the definition:

1. Can we ask for security against all adversaries (when $n \gg \lambda$)?

   No! Consider <u>inefficient</u> adversary that outputs 1 if t is the image of G and 0 otherwise.

   - $W_0 = 1$
   - $W_1 = \Pr[t \xleftarrow{R} \{0,1\}^n : \exists s \in \{0,1\}^\lambda : G(s) = t] = \frac{1}{2^{n-\lambda}}$

   $\Big\}$ $\text{PRGAdv}[A,G] = 1 - \frac{1}{2^{n-\lambda}} \approx 1$ if $n \gg \lambda$

2. Can the output of a PRG be biased (e.g., first bit of PRG output is 1 w.p. $\frac{2}{3}$)?

   No! Consider <u>efficient</u> adversary that outputs 1 if first bit of challenge is 1.

   - $W_0 = \frac{2}{3}$
   - $W_1 = \frac{1}{2}$

   $\Big\}$ $\text{PRGAdv}[A,G] = \frac{1}{6}$ <u>NOT NEGLIGIBLE!</u>

   More generally, no efficient statistical test can distinguish output of a secure PRG from random.

3. Can the output of a PRG be predictable (e.g., given first 10 bits, predict the 11^{th} bit)?

   No! If the bits are predictable w.p. $\frac{1}{2} + \varepsilon$, can distinguish with advantage ε (since random string is unpredictable)

   <u>In fact</u>: unpredictable $\Rightarrow$ pseudorandom

<u>Take-away</u>: A secure PRG has the same statistical properties as the one-time pad to any efficient adversary.

$\Longrightarrow$ Should be able to use it in place of one-time pad to obtain a <u>secure</u> encryption scheme (against efficient adversaries)

Exercising the definition: we will now consider an example of proving security of a PRG

<u>Theorem</u> Suppose $G: \{0,1\}^\lambda \to \{0,1\}^n$ is a secure PRG. Then, the function $G'(s) := G(s) \oplus 1^n$ is also a secure PRG.

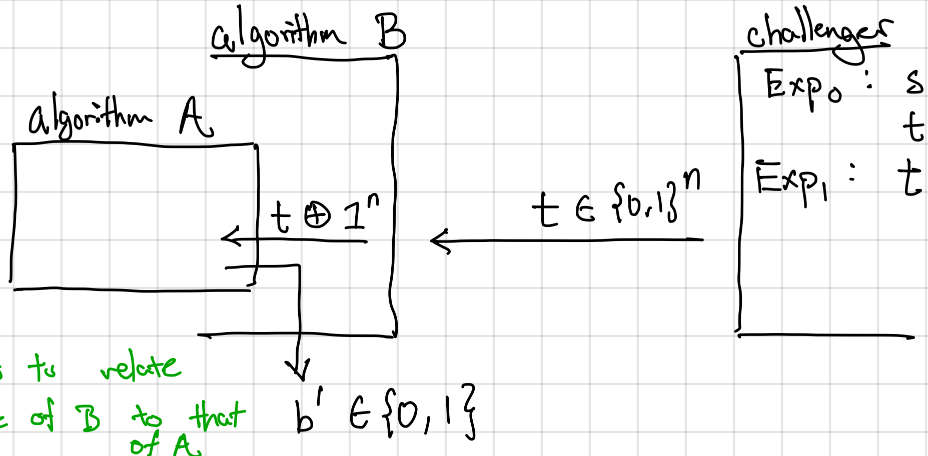
This is a conditional statement! We will discuss this more in the coming lectures.

<u>Proof.</u> To prove this directly seems difficult: must show <u>non-existence</u> of an adversary.

Instead, we consider the contrapositive:

"If G' is <u>not</u> a secure PRG, then G is <u>not</u> a secure PRG"

Suppose G' is not secure. Namely, there exists an efficient adversary A that breaks security of G' with non-negligible advantage ε. We use A to construct a new adversary B that breaks security of G:



algorithm B basically runs A under the hood

$\hookrightarrow$ our goal is to relate the advantage of B to that of A.

In Exp_0, algorithm $\mathcal{B}$ invokes algorithm $\mathcal{A}$ on the string $G(s) \oplus 1^n$ where $s \xleftarrow{R} \{0,1\}^\lambda$ is random. This is precisely the distribution of Exp_0 for $\mathcal{A}$. Thus,

$$W_0 = \Pr[\mathcal{B} \text{ outputs } 1 \text{ in } \text{Exp}_0] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in } \text{Exp}_0]$$

In Exp_1, algorithm $\mathcal{B}$ invokes algorithm $\mathcal{A}$ on the string $t \oplus 1^n$ where $t \xleftarrow{R} \{0,1\}^n$ is uniformly random. The distribution of $t \oplus 1^n$ is still uniform:

$$\forall u \in \{0,1\}^n : \Pr[t \xleftarrow{R} \{0,1\}^n : t \oplus 1^n = u]$$

$$= \Pr[t \xleftarrow{R} \{0,1\}^n : t = u \oplus 1^n] = \frac{1}{2^n}$$

This means

$$W_1 = \Pr[\mathcal{B} \text{ outputs } 1 \text{ in } \text{Exp}_1] = \Pr[\mathcal{A} \text{ outputs } 1 \text{ in } \text{Exp}_1]$$

We conclude then that

$$\text{PRGAdv}[\mathcal{B}, G] = |W_0 - W_1|$$

$$= \big|\Pr[\mathcal{A} \text{ outputs } 1 \text{ in } \text{Exp}_0] - \Pr[\mathcal{A} \text{ outputs } 1 \text{ in } \text{Exp}_1]\big|$$

$$= \varepsilon,$$

which is non-negligible by assumption. This proves the contrapositive.

The above proof is an example of a security <u>reduction</u>. We show how to reduce the task of breaking G to that of breaking G'. This means an attack on G' implies an attack on G. Correspondingly, if G is secure (i.e., no efficient attacks succeed with non-negligible probability), then the same holds for G'.

Refer to the posted notes on the course website as well as the textbook for more examples. We will see more reductions throughout the course as well.