Do PRGs exist? We don't know! More difficult problem than resolving P vs. NP!

However, it is not hard to see that if PRGs exist, then $P \neq NP$. [Try proving this yourself]

↳ What we can say is that if "one-way functions" (OWF) exist, then there exists a PRG that stretches the seed by 1 bit (e.g., λ-bit seed → $(\lambda+1)$-bit string)

"one-way functions": function that is "easy" to compute but "hard" to invert

↳ will define more formally later in the course

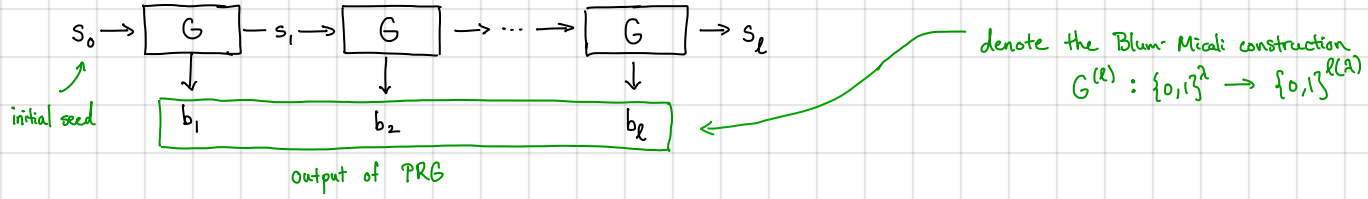
a PRG is an example of such a function

given $s \in \{0,1\}^\lambda$, evaluating $G(s) \in \{0,1\}^n$ is easy

given $G(s) \in \{0,1\}^n$ for random $s \in \{0,1\}^\lambda$, computing s is hard (why?)

But what if we want PRGs with longer stretch? For example, can we build PRGs with stretch $\ell(\lambda) = \text{poly}(\lambda)$ for arbitrary polynomials?

Blum-Micali PRG: Suppose $G: \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a secure PRG. We build a PRG with stretch $\ell(\lambda) = \text{poly}(\lambda)$ as follows:



Why is this constructing a secure PRG?

↳ Intuitively, if s_0 is uniformly random, then $G(s_0) = (b_1, s_1)$ is uniformly random so we can feed s_1 into the PRG and take b_1 as the first output bit of the PRG ⇒ iterate until we have ℓ output bits

Theorem. If $G: \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is a secure PRG, then the Blum-Micali generator $G^{(\ell)}: \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ is also a secure PRG for all $\ell = \text{poly}(\lambda)$.

Proof. Consider the following experiments:

Experiment H_0: Sample $s_0 \xleftarrow{R} \{0,1\}^\lambda$ and adversary is given $G^{(\ell)}(s_0)$

Experiment H_1: Sample $t \xleftarrow{R} \{0,1\}^{\ell(\lambda)}$ and adversary is given t

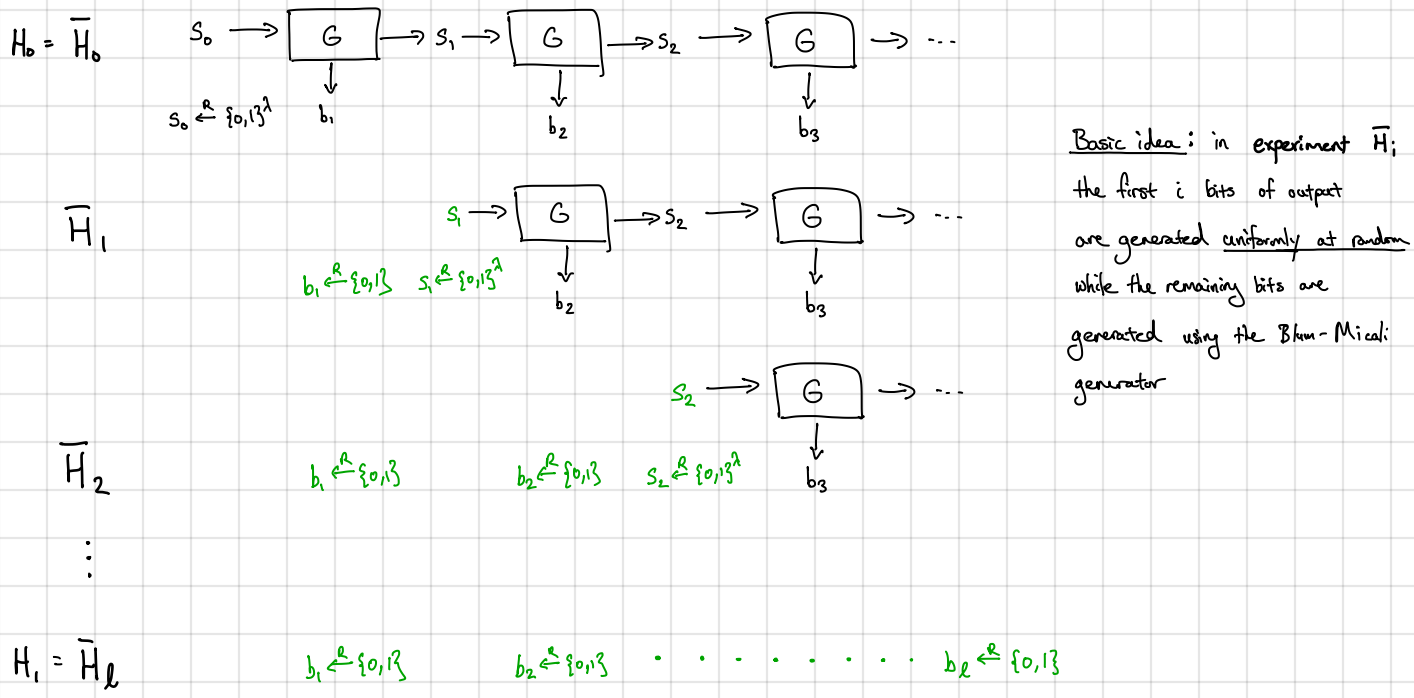
For an adversary A, define

$W_0 := \Pr[A \text{ outputs } 1 \text{ in } H_0]$

$W_1 := \Pr[A \text{ outputs } 1 \text{ in } H_1]$

Goal: Show that if G is secure, then for all efficient adversaries A, $|W_0 - W_1| = \text{negl}(\lambda)$.

We will use a "hybrid" argument. Specifically, we first define a sequence of intermediate experiments, where each adjacent pair of experiments is easy to reason about (i.e., directly reduces to security of G)



Basic idea: in experiment $\bar{H}_i$ the first i bits of output are generated uniformly at random while the remaining bits are generated using the Blum-Micali generator

In each experiment, adversary is given the sequence of bits $b_1 b_2 \cdots b_\ell$

Let A be an efficient distinguisher. Define $\bar{W}_i := \Pr[A \text{ outputs } 1 \text{ in experiment } \bar{H}_i]$

$$\begin{aligned}
\text{Then, } \mathrm{PRGAdv}[A,G] &= |W_0 - W_\ell| \\
&= |\bar{W}_0 - \bar{W}_\ell| \qquad \text{(by definition)} \\
&= |\bar{W}_0 - \bar{W}_1 + \bar{W}_1 - \bar{W}_2 + \cdots + \bar{W}_{\ell-1} - \bar{W}_\ell| \\
&\le |\bar{W}_0 - \bar{W}_1| + |\bar{W}_1 - \bar{W}_2| + \cdots + |\bar{W}_{\ell-1} - \bar{W}_\ell| \qquad \text{(by triangle equality)}
\end{aligned}$$

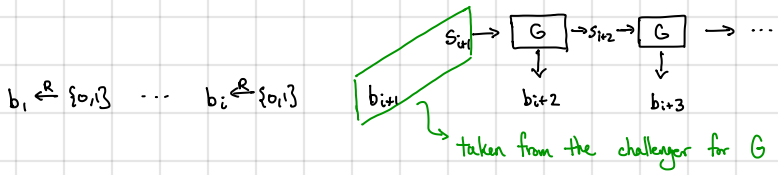
Claim. If G is a secure PRG, then for all efficient adversaries A, $|\bar{W}_i - \bar{W}_{i+1}| = \mathrm{negl}(\lambda)$.

Proof. We will show the contrapositive: if A can distinguish experiments $\bar{H}_i$ and $\bar{H}_{i+1}$, then A can break pseudorandomness of G.

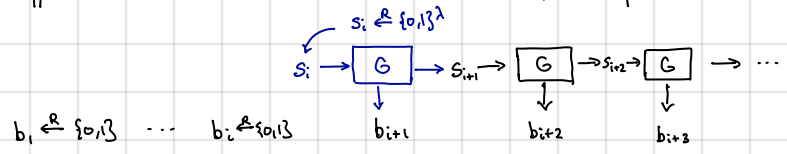
Suppose $|\bar{W}_i - \bar{W}_{i+1}| = \varepsilon$. We use A to build a distinguisher B for G. Algorithm B works as follows:

1. On input a string $z \in \{0,1\}^{\lambda+1}$, algorithm B parses z as (b_{i+1}, s_{i+1}) where $b_{i+1} \in \{0,1\}$ and $s_{i+1} \in \{0,1\}^\lambda$
2. Sample $b_1, \ldots, b_i \xleftarrow{R} \{0,1\}$.
3. Compute $b_{i+2}, \ldots, b_\ell$ using Blum-Micali with seed s_{i+1}. Give $b_1 \cdots b_\ell$ to A and output whatever A outputs.

In pictures:

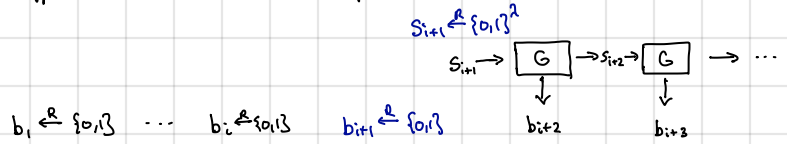


Two possibilities: 1. Suppose $z = G(s_i)$ for some $s_i \xleftarrow{R} \{0,1\}^\lambda$. Then, above picture looks like this:



In this case, $b_1, \ldots, b_\ell$ is distributed exactly as in experiment $\overline{H}_i$ and so A outputs 1 with prob. $\overline{W}_i$

2. Suppose $z \xleftarrow{R} \{0,1\}^{2\lambda+1}$. Then above picture looks like this:



In this case, $b_1, \ldots, b_\ell$ is distributed exactly as in experiment $\overline{H}_{i+1}$ and so A outputs 1 with prob. $\overline{W}_{i+1}$

Very important to argue that B "simulates" the correct views for A. Otherwise, behavior of A is unknown!

Thus, $\text{PRGAdv}[B,G] = |\overline{W}_i - \overline{W}_{i+1}|$
$= \varepsilon$ since B outputs whatever A outputs

Since B is efficient (assuming A is efficient), by security of G, $\text{PRGAdv}[B,G] = \text{negl}(\lambda)$. Thus, $\varepsilon = |p_i - p_{i+1}| = \text{negl}(\lambda)$, and the claim follows. ■

To complete the proof of the main theorem, we have that

$$|\overline{W}_0 - \overline{W}_\ell| \le |\overline{W}_0 - \overline{W}_1| + \cdots + |\overline{W}_{\ell-1} - \overline{W}_\ell|$$
$$\le \ell \cdot \text{negl}(\lambda)$$
$$= \text{negl}(\lambda) \quad \text{since } \ell = \text{poly}(\lambda).$$

Proof strategy recap: 1. Hybrid arguments: to argue indistinguishability of a pair of distributions, begin by identifying a simple set of intermediate distributions, and argue that each pair of adjacent distributions is indistinguishable

2. Security reduction (proof by contrapositive): To show a statement of the form "If X is secure, then Y is secure," show instead the the statement "If Y is not secure, then X is not secure." In the proof, show that if there exists an adversary for Y (i.e. Y is not secure), then there exists an adversary for X.
↳ When constructing this adversary, it is important to show that it simulates the correct distribution of inputs to the underlying adversary (i.e., this is essentially showing correctness of the reduction algorithm)

Stream ciphers in practice:
- Salsa20 (2005) ⇝ ChaCha (2008)
↳ core design maps 256-bit key, 64-bit nonce, 64-bit counter onto a 512-bit output

enables using same key (and different nonces) to encrypt multiple messages (will discuss later)

allows random access into the stream

Design is more complex:
- relies on a sequence of rounds
- each round consists of 32-bit additions, xors, and bit-shifts

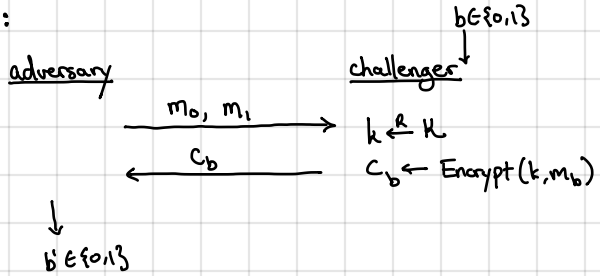
↳ very fast even in software (4-14 CPU cycles / output byte) — used to encrypt TLS traffic between Android and Google services

Recall: the one-time pad is not reusable (i.e., the two-time pad is totally broken)

NEVER REUSE THE KEY TO A STREAM CIPHER!

But wait... we "proved" that a stream cipher was secure, and yet, there is an attack?

Recall security game:



Observe: adversary only sees one ciphertext
key is only used once

⇒ Security in this model says nothing about multiple messages / ciphertexts

Problem: If we want security with multiple ciphertexts, we need a different or stronger definition (CPA security)

Reusable security: security against chosen-plaintext attacks (CPA-security)

(adversary does not just passively observe, it can choose the messages to be encrypted!)

↳ semantic security should hold even if adversary sees multiple encrypted messages of its choosing

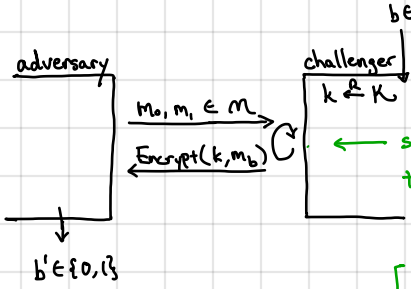
↳ captures many settings where adversary might know the message that is encrypted (e.g., predictable headers or site content in web traffic) or be able to influence it (e.g., client replies to an email sent by adversary)

↳ goal is to capture as broad of a range of attacks as possible

Definition: An encryption scheme $\Pi_{SE} = (\text{Encrypt}, \text{Decrypt})$ is secure against chosen-plaintext attacks (CPA-secure) if for all efficient adversaries A:

$$\text{CPAAdv}[A, \Pi_{SE}] = \left| \Pr[W_0 = 1] - \Pr[W_1 = 1] \right| = \text{negl}.$$

where W_b ($b \in \{0,1\}$) is the output of the following experiment:



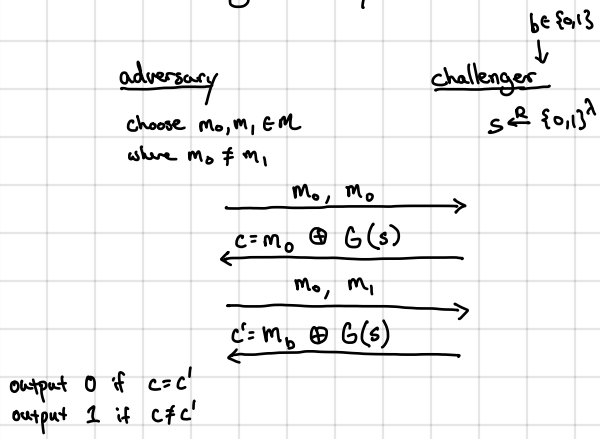
← same idea as in original semantic security game, but allow adversary to make encryption queries (also called a "left-or-right" oracle)

output of experiment W_b

[Adversary's goal is to guess which of m_0 or m_1 was encrypted, given access to an encryption oracle (i.e., adversary gets to see encryptions of messages of its choice.]

Claim. A stream cipher is not CPA-secure.

Proof. Consider the following adversary:



$\Pr[b' = 1 \mid b = 0] = 0$ since $c' = m_0 \oplus G(s) = c$

$\Pr[b' = 1 \mid b = 1] = 1$ since $c' = m_1 \oplus G(s) \neq c$

$\Rightarrow \text{CPAAdv}[A, \Pi_{SE}] = 1$

Observe: Above attack works for any deterministic encryption scheme.

$\Rightarrow$ CPA-secure encryption must be randomized!

$\Rightarrow$ To be reusable, cannot be deterministic. Encrypting the same message twice should not reveal that identical messages were encrypted.

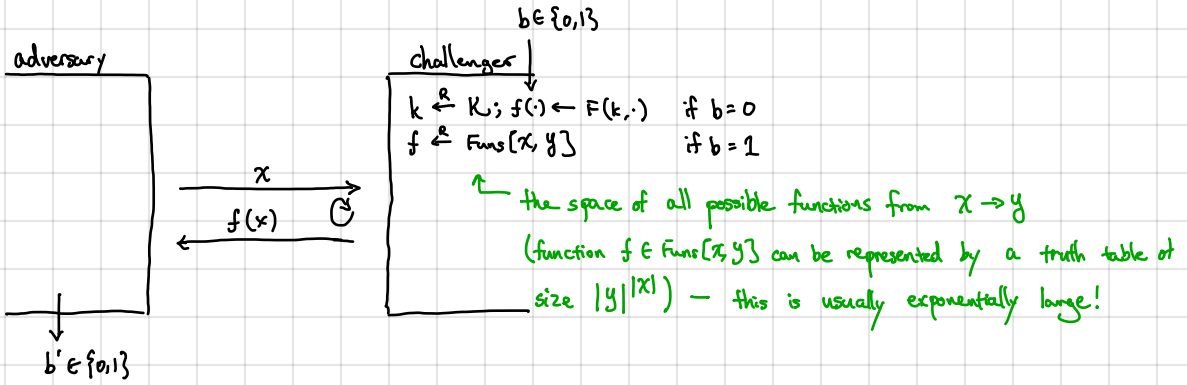
To build a CPA-secure encryption scheme, we will use a "block cipher"

- Block cipher is an invertible keyed function that takes a block of n input bits and produces a block of n output bits
- Examples include 3DES (key size 168 bits, block size 64 bits)
  AES (key size 128 bits, block size 128 bits)

Will define block ciphers abstractly first: pseudorandom functions (PRFs) and pseudorandom permutations (PRPs) (block ciphers)

↳ General idea: PRFs behave like random functions
PRPs behave like random permutations

Definition. A function $F: K \times X \to Y$ with key-space K, domain X, and range Y is a pseudorandom function (PRF) if for all efficient adversaries A, $|W_0 - W_1| = \text{negl.}$, where W_b is the probability the adversary outputs 1 in the following experiment:



↑ the space of all possible functions from $X \to Y$

(function $f \in \text{Funs}[X, Y]$ can be represented by a truth table of size $|Y|^{|X|}$) — this is usually exponentially large!

$$\text{PRFAdv}[A, F] = |W_0 - W_1| = |\Pr[A \text{ outputs } 1 \mid b = 0] - \Pr[A \text{ outputs } 1 \mid b = 1]|$$

Intuitively: input-output behavior of a PRF is indistinguishable from that of a random function (to any computationally-bounded adversary)

3DES:	$\{0,1\}^{168} \times \{0,1\}^{64} \to \{0,1\}^{64}$	$\lvert K \rvert = 2^{168}$	$\lvert \text{Funs}[X,Y] \rvert = (2^{64})^{(2^{64})}$	} space of random functions is exponentially-larger than key-space!
AES:	$\{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$	$\lvert K \rvert = 2^{128}$	$\lvert \text{Funs}[X,Y] \rvert = (2^{128})^{(2^{128})}$	

Definition: A function $F: K \times X \to X$ is a pseudorandom permutation (PRP) if

- for all keys k, $F(k, \cdot)$ is a permutation and moreover, there exists an efficient algorithm to compute $F^{-1}(k, \cdot)$:

$$\forall k \in K : \forall x \in X : F^{-1}(k, F(k,x)) = x$$

- for $k \xleftarrow{R} K$, the input-output behavior of $F(k, \cdot)$ is computationally indistinguishable from $f(\cdot)$ where $f \xleftarrow{R} \text{Perm}[X]$ and $\text{Perm}[X]$ is the set of all permutations on X (analogous to PRF security)

Note: a block cipher is another term for PRP (just like stream ciphers are PRGs)