- Modern stream ciphers (eSTREAM project: 2004-2008)
  - 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 <u>multiple</u> messages (will discuss later)

      allows <u>random</u> 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

<u>Recall</u>: 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:

<u>adversary</u>		<u>challenger</u> ($b \in \{0,1\}$)
	m_0, m_1 →	$k \xleftarrow{R} \mathcal{K}$
	← c_b	$c_b \leftarrow \text{Encrypt}(k, m_b)$
↓ $b' \in \{0,1\}$		

<u>Observe</u>: adversary only sees <u>one</u> ciphertext

key is only used <u>once</u>

⟹ Security in this model says <u>nothing</u> about multiple messages / ciphertexts

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

<u>Reusable security</u>: security against chosen-plaintext attacks $\boxed{\text{(CPA-security)}}$

- ↳ semantic security should hold even if adversary sees multiple encrypted messages of <u>its choosing</u>

  adversary does not just passively observe, it can choose the messages to be encrypted!
- ↳ 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 $\mathcal{A}$:

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

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

adversary — challenger ($b \in \{0,1\}$; $k \xleftarrow{R} \mathcal{K}$)

$m_0, m_1 \in \mathcal{M}$ →

← $\text{Encrypt}(k, m_b)$

$b' \in \{0,1\}$

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:

adversary — challenger ($b \in \{0,1\}$; $s \xleftarrow{R} \{0,1\}^\lambda$)

choose $m_0, m_1 \in \mathcal{M}$ where $m_0 \neq m_1$

m_0, m_0 →

← $c = m_0 \oplus G(s)$

m_0, m_1 →

← $c' = m_b \oplus G(s)$

output 0 if $c = c'$
output 1 if $c \neq c'$

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

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

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

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.

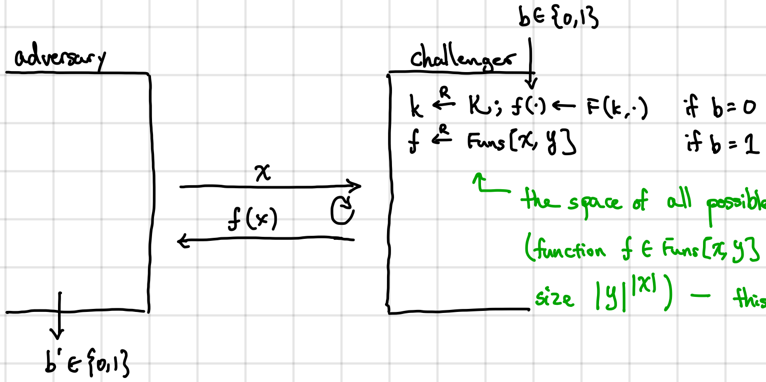
To build a CPA-secure encryption scheme, we will use a "block cipher"

- 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 $\underbrace{\text{pseudorandom permutations (PRPs)}}_{\text{block ciphers}}$

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

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



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

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

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

3DES: $\{0,1\}^{168} \times \{0,1\}^{64} \to \{0,1\}^{64}$ $\quad |\mathcal{K}| = 2^{168}$ $\quad |\text{Funs}[\mathcal{X},\mathcal{Y}]| = (2^{64})^{(2^{64})}$

AES: $\{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ $\quad |\mathcal{K}| = 2^{128}$ $\quad |\text{Funs}[\mathcal{X},\mathcal{Y}]| = (2^{128})^{(2^{128})}$

} space of random functions is exponentially-larger than key-space!

Definition: A function $F: \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ 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 $F^{-1}(k,\cdot)$:

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

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

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

Observe that a block cipher can be used to construct a PRG:

$F : \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher

Define $G : \{0,1\}^\lambda \to \{0,1\}^{\ell n}$ as

$$G(k) = F(k,1) \,\|\, F(k,2) \,\|\, \cdots \,\|\, F(k,\ell)$$

← this stream cipher allows random access!

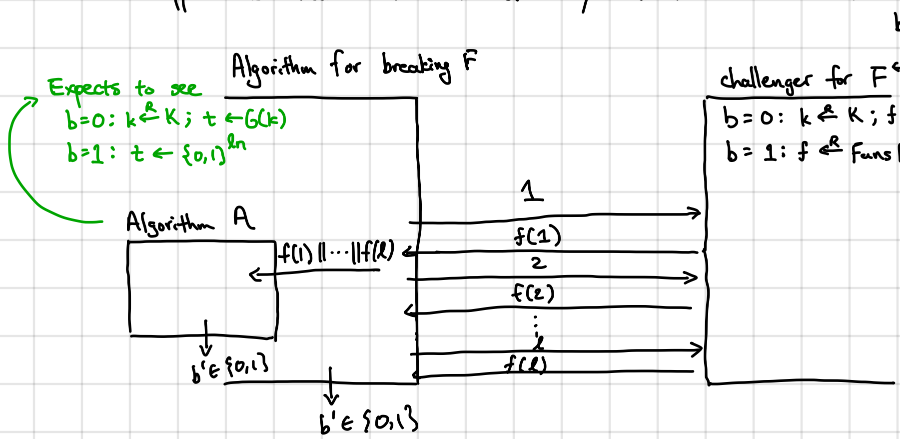
↑ string concatenation ↑ write input as an n-bit string (just require that $n > \log \ell$)

we said PRP above (will revisit this)

Theorem. If F is a secure PRF, then G is a secure PRG.

Proof. As usual, we show the contrapositive: if G is not a secure PRG, then F is not a secure PRF.

Suppose we have efficient adversary A for G. We use A to build adversary for F:



1. If $\ell = \text{poly}$, then B is efficient
2. If $b=0$: B sends $G(k)$ to A where k is a uniformly random key

   If $b=1$: B sends uniformly random string (f is random function) to A
3. $\text{PRFAdv}[B,F] = \left|\Pr[b'=1 \mid b=0] - \Pr[b'=1 \mid b=1]\right|$

$= \left|\Pr[A \text{ outputs } 1 \mid b=0] - \Pr[A \text{ outputs } 1 \mid b=1]\right|$

$= \text{PRGAdv}[A,G]$

which is non-negligible by assumption.

But... we used a block cipher (PRP) in our construction above. Does the proof still go through?

Not quite...

for a random function, $f(1) = f(2)$ with probability $\frac{1}{2^n}$

for a random permutation, $f(1) = f(2)$ with probability 0

} but 2^{-n} might be very very small... adversary won't notice unless it sees a "collision" [i.e., two values x,y where $f(x) = f(y)$]

PRF Switching Lemma. Let $F : K \times X \to X$ be a secure PRP. Then, for any Q-query adversary A:

$$\left|\text{PRPAdv}[A,F] - \text{PRFAdv}[A,F]\right| \le \frac{Q^2}{2|X|}$$

Proof Idea. Adversary essentially cannot tell the difference unless it sees a collision. If there is no collision, then it is just seeing random values. How many queries before there is a collision? Birthday paradox: $Q \sim \sqrt{|X|}$

Take-away: If $|X|$ is large (e.g., exponential), then we can use a PRP as a PRF.

- 3DES: $n = 64$ so $|X| = 2^{64}$ [if adversary makes $\ll 2^{32}$ queries, then can use it as a PRF]
- AES: $n = 128$ so $|X| = 2^{128}$ [if adversary makes $\ll 2^{64}$ queries, then can use it as a PRF]