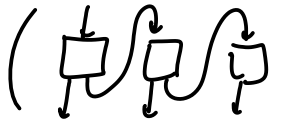
Pseudorandom Functions (PRF)

Last Time: Pseudorandom Generators (PRGs)

short input ⟶ long output

security: when input (seed) chosen uniformly, output "looks uniform"

Can extend a PRG ( .... idea)

but no matter what, output length is polynomial function of input length (e.g. $G: \{0,1\}^\lambda \to \{0,1\}^{\lambda^2}$)

let's get greedy! from a λ-bit seed, get 2^λ bits that "look uniform"

Change the game: poly-time algo doesn't have time to read/write 2^λ bits

instead, ask for RANDOM ACCESS to long 2^λ-length data

given seed k, index x, compute x^{th} ~~bit~~ block of this long string

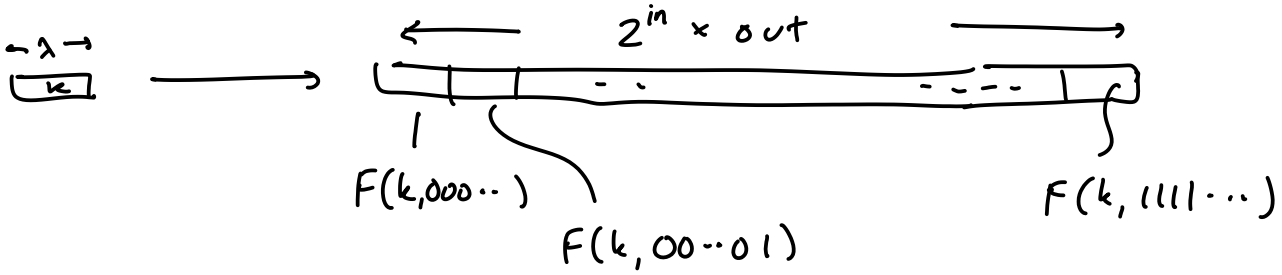
could be a poly-time computation

Def: A pseudorandom function $F: \{0,1\}^{\lambda} \times \{0,1\}^{in} \to \{0,1\}^{out}$

seed/key — "index" — "block"

Idea: Given $k \leftarrow \{0,1\}^{\lambda}$, it defines a very long string

$F(k,000\cdots) \,||\, F(k,00\cdots01) \,||\, \cdots \,||\, F(k,11\cdots)$



Security: random access to this huge string "looks like" random access to uniformly chosen string

formal security def: F is a secure PRF if:

$k \leftarrow \{0,1\}^{\lambda}$
QUERY($x \in \{0,1\}^{in}$):
  return $F(k,x)$
	$\approx\approx$	cache = empty dict.
		QUERY(x):
		  if cache[x] undef:
		  cache[x] $\leftarrow \{0,1\}^{out}$
		  return cache[x]

Why PRF? fix k, (currying)

then $F(k,\cdot): \{0,1\}^{in} \to \{0,1\}^{out}$

$F(k,\cdot)$ "looks like" randomly selected func $\{0,1\}^{in} \to \{0,1\}^{out}$

Attacks!

A bad PRF:

$F(k, x) = G(k) \oplus x$ where G is a PRG

Hint: break F not G, break F even if G is an awesome PRG

Idea: this is (pseudo) OTP, so make 2 queries, both will use same k (two-time pad?)

distinguisher

A:

choose $x_1 \neq x_2$ arbitrarily

$z_1 = \text{QUERY}(x_1)$

$z_2 = \text{QUERY}(x_2)$

return $z_1 \oplus z_2 \stackrel{?}{=} x_1 \oplus x_2$

in PRF "real"

$z_1 = G(k) \oplus x_1$

$z_2 = G(k) \oplus x_2$

$\Pr[\text{output TRUE}] = 1$

in PRF "rand"

$z_1 \leftarrow \{0,1\}^{out}$

$z_2 \leftarrow \{0,1\}^{out}$

$\Pr[\text{output TRUE}] = \Pr[z_2 = z_1 \oplus x_1 \oplus x_2]$

$= 1/2^{out}$