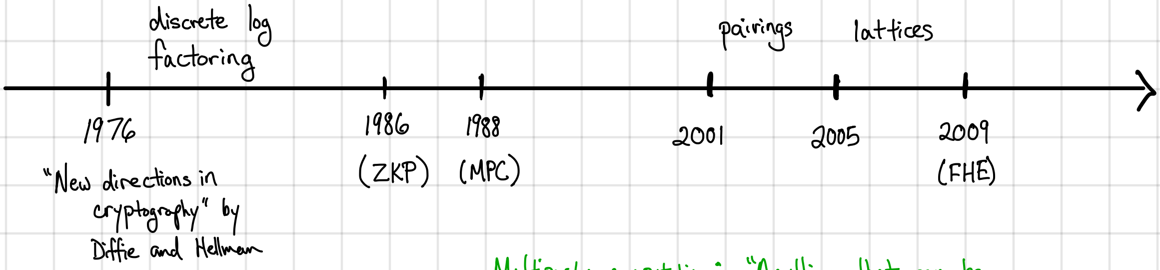
CS 6501 Week 1: Definitions and Foundations

Instructor: David Wu (dwu4@virginia.edu)
TA: Suya (fs5xz@virginia.edu)

What this course is all about:

Zero-knowledge proofs: How do you convince someone that a statement is true but not reveal anything more about the statement?
What does it even mean to "know" something?

Fully homomorphic encryption: Can we compute on encrypted data (e.g., preserve full confidentiality of the data and yet, perform arbitrary computations over the hidden data?



Public-key cryptography:
How can two users who have never met communicate securely over a public and untrusted network?

Multiparty computation: "Anything that can be computed with the help of a trusted party can be computed without one!"

Cryptography started as a means of protecting communication. Has now evolved to also protect computation.

We are at an "inflection" point in the development of cryptography
1. Real, large-scale deployment of "fancy" cryptography (i.e. beyond public-key cryptography)
   ↳ Systems like Zcash (relies on pairings-based cryptography and zero-knowledge proof systems)
2. Potential threat of quantum computers requires re-thinking much of the existing public-key cryptography
   - Google recently ran pilot project implementing ring-LWE based key-exchange into Chrome (alongside traditional Diffie-Hellman)
   - Ongoing NIST competition to standardize new post-quantum cryptography (expect 5-7 years to converge on new standards)

Course objectives: 1. Provide broad survey of modern cryptography (from a theory-focused lens) ← so we will be starting with foundations
2. Prepare you for research in cryptography
↳ Will likely offer an introduction to applied cryptography course next year

↳ Course will be fast-paced at the beginning, but should be self-contained.
More thorough treatment of symmetric crypto and public-key crypto (especially how to use them properly in systems) will be covered in the applied crypto course.

Logistics and administrivia: Course website: https://www.cs.virginia.edu/dwu4/courses/sp19 ← refer here for late day policy, homework submission instructions, office hours, notes, ...
See Piazza for announcements, Gradescope for homework submission
Anonymous feedback form also available ← please provide feedback!
Course consists of 5 problem sets, weighted equally; no exams or projects
Course TA: Suya [see website for contact information]

Foundations of Modern Cryptography

Modern cryptography is study of hardness: some task should be "easy" for an honest user, but "difficult" for an adversary

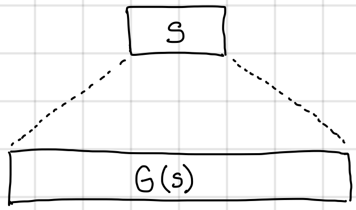
e.g., if Alice encrypts a message to Bob, it should be easy for Alice to encrypt the message and for Bob to decrypt; however, should be difficult for eavesdropper who intercepts the message to decrypt

How do we model this mathematically?

basic building block of symmetric cryptography (will see how to use PRGs/PRFs/PRPs to build encryption schemes, message authentication codes next week)

Consider the notion of a pseudorandom generator (PRG):

- A PRG takes a short seed s and expands it to a much longer random string $G(s)$.
- We will see (next week) that PRGs (and related primitives) are very useful for constructing encryption schemes, message integrity mechanisms, and more.



security parameter — stretch of the PRG — running time is polynomial in the input length

Definition. A function $G: \{0,1\}^{\boxed{\lambda}} \to \{0,1\}^{\boxed{\ell(\lambda)}}$ is a secure pseudorandom generator (PRG) if for all $\boxed{\text{efficient}}$ algorithms $\boxed{A}$:

$$\left| \Pr[s \xleftarrow{R} \{0,1\}^\lambda : A(G(s)) = 1] - \Pr[t \xleftarrow{R} \{0,1\}^{\ell(\lambda)} : A(t) = 1] \right| = \boxed{\text{negl}(\lambda)}$$

without loss of generality, assume A just outputs a single bit

we view A as a "distinguisher": on input a string of length $\ell(\lambda)$, guesses whether it is the output of a PRG or if it is truly random — notice that typically $\ell(\lambda) \gg \lambda$ so $2^\lambda \ll 2^{\ell(\lambda)}$ (remarkable that PRGs plausibly exist!)

a function $f(\lambda)$ is negligible if $f(\lambda) = o(1/\lambda^c)$ for all $c \in \mathbb{N}$ (i.e., the function f is smaller than all polynomials in λ)

Intuitively: the outputs of a PRG on a random seed looks indistinguishable from a truly random string (assuming adversary does not see the seed)

Alternative characterization in terms of distributions: for an adversary A, define the following random variables:

$W_0 : \Pr[s \xleftarrow{R} \{0,1\}^\lambda : A(G(s)) = 1]$ "pseudorandom" distribution

$W_1 : \Pr[t \xleftarrow{R} \{0,1\}^{\ell(\lambda)} : A(t) = 1]$ "truly random" distribution

$\text{PRGAdv}[A, G] := |W_0 - W_1|$

Definition. A function $G: \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ is a secure PRG if for all efficient adversaries A, $\text{PRGAdv}[A,G] = \text{negl}(\lambda)$.

PRG security definition requires that two distribution ensembles look indistinguishable to any computationally-bounded adversary. More generally, we can write:

Definition. Let $\lambda \in \mathbb{N}$ be a security parameter. Let $\mathcal{D}_1 = \{\mathcal{D}_{1,\lambda}\}_{\lambda \in \mathbb{N}}$ and $\mathcal{D}_2 = \{\mathcal{D}_{2,\lambda}\}_{\lambda \in \mathbb{N}}$ be two collections (ensembles) of distributions indexed by λ. Then, $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable (denoted $\mathcal{D}_1 \overset{c}{\approx} \mathcal{D}_2$) if for all efficient adversaries A,

$$\left| \Pr[x_1 \leftarrow \mathcal{D}_{1,\lambda} : A(1^\lambda, x_1) = 1] - \Pr[x_2 \leftarrow \mathcal{D}_{2,\lambda} : A(1^\lambda, x_2) = 1] \right| = \text{negl}(\lambda)$$

the adversary is given 1^λ (the all-ones string of length λ); this allows the running time of the algorithm to be $\text{poly}(\lambda)$ (the draws from $\mathcal{D}_{1,\lambda}$ and $\mathcal{D}_{2,\lambda}$ may be much shorter than λ)

Intuitively: no efficient adversary can tell a sample from $\mathcal{D}_1$ from a sample from $\mathcal{D}_2$

PRG security definition: A function $G: \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ is a secure PRG if $\{s \xleftarrow{R} \{0,1\}^\lambda : G(s)\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} \{t \xleftarrow{R} \{0,1\}^{\ell(\lambda)} : t\}$.

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

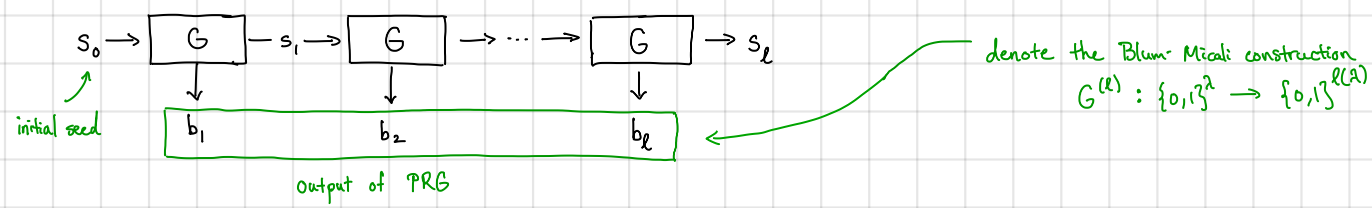
However, it is not hard to see that if PRGs exist, then P ≠ NP. [Try proving this yourself]

↳ What we can say is that if one-way functions (OWF) exist, then there exists a PRG where $\ell(\lambda) = \lambda + 1$ (i.e., a PRG with a 1-bit stretch)

[We will explore this more thoroughly on HW1]

But what if we want PRGs with <u>longer</u> stretch? For example, can we build PRGs with stretch $\ell(\lambda) = \mathrm{poly}(\lambda)$ 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 $\ell(\lambda) = \mathrm{poly}(\lambda)$ as follows:



Why is this constructing a secure PRG?

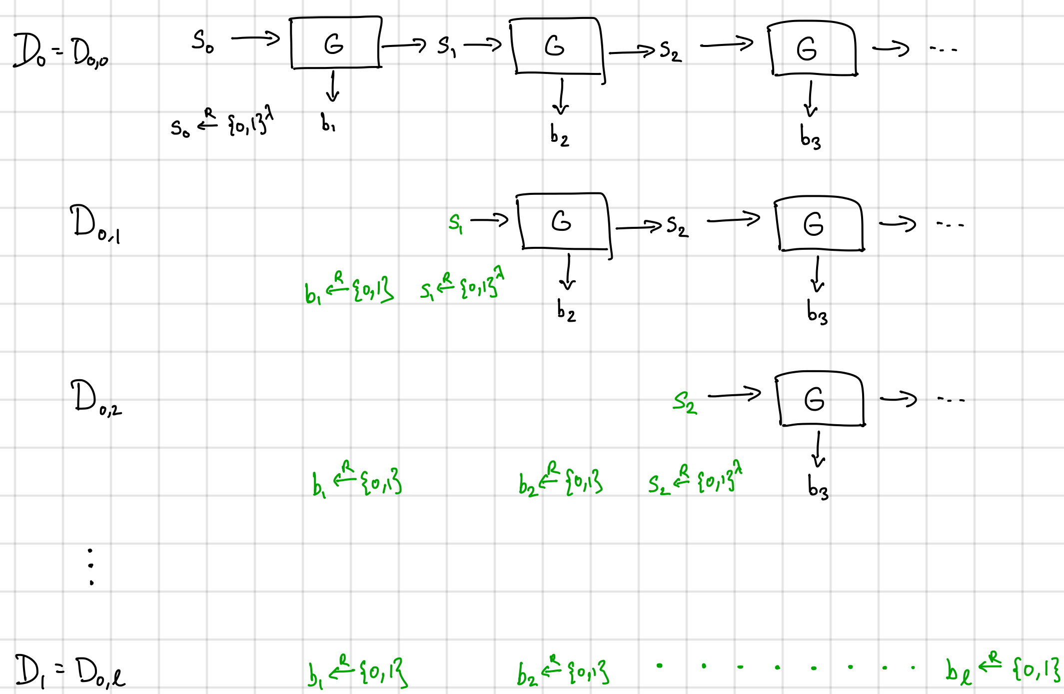
↳ Intuitively, if s_0 is uniformly random, then $G(s_0) = (b_1, s_1)$ is uniformly random so we can feed s_1 into the PRG and take b_1 as the first output bit of the PRG ⟹ iterate until we have ℓ output bits

<u>Theorem.</u> 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 $\ell = \mathrm{poly}(\lambda)$.

<u>Proof.</u> Our goal is to show that the following distributions are computationally indistinguishable:

$$D_0 = \{ s_0 \xleftarrow{R} \{0,1\}^\lambda : G^{(\ell)}(s_0) \}$$
$$D_1 = \{ t \xleftarrow{R} \{0,1\}^{\ell(\lambda)} : t \}$$

We will use a "hybrid" argument. Specifically, we first define a <u>sequence</u> of intermediate distributions:



<u>Basic idea</u>: in distribution $D_{0,i}$, the first i bits of output are generated <u>uniformly at random</u> while the remaining bits are generated using the Blum-Micali generator

Let A be an efficient distinguisher. Define $p_i = \Pr[x \xleftarrow{R} D_{0,i} : A(x) = 1]$.

Our objective is to show that $\mathrm{PRGAdv}[A, G] = |p_0 - p_\ell| = \mathrm{negl}(\lambda)$.

Proof (cont.). We use the triangle inequality:

$$\mathrm{PRGAdv}[A,G] = |p_0 - p_\ell| = |p_0 - p_1 + p_1 - p_2 + \cdots + p_{\ell-1} - p_\ell|$$
$$\leq |p_0 - p_1| + |p_1 - p_2| + \cdots + |p_{\ell-1} - p_\ell|$$

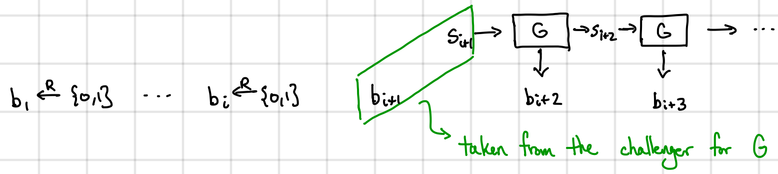
Claim. If G is a secure PRG, then for all efficient adversaries A, $|p_i - p_{i+1}| = \mathrm{negl}(\lambda)$.

Proof. We will show the contrapositive: if A can distinguish distributions $D_{0,i}$ and $D_{0,i+1}$, then A can break pseudorandomness of G.

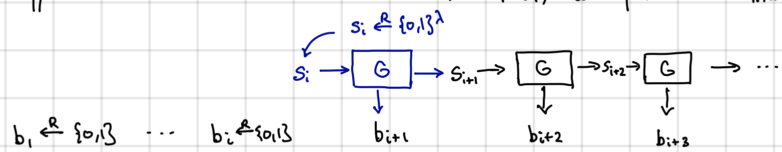
Suppose $|p_i - p_{i+1}| = \varepsilon$. We use A to build a distinguisher B for G. Algorithm B works as follows:

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:

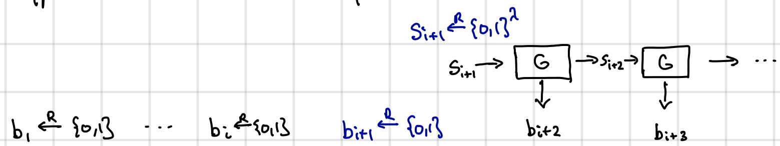


Two possibilities: 1. Suppose $z = G(s_i)$ for some $s_i \xleftarrow{R} \{0,1\}^\lambda$. Then, above picture looks like this:



In this case, $b_1, \ldots, b_\ell$ is distributed exactly as in distribution $D_{0,i}$ and so A outputs 1 with prob. p_i

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



In this case, $b_1, \ldots, b_\ell$ is distributed exactly as in distribution $D_{0,i+1}$ and so A outputs 1 with prob. p_{i+1}

Thus, $\mathrm{PRGAdv}[B,G] = \left| \Pr[s \xleftarrow{R} \{0,1\}^\lambda : B(G(s))] - \Pr[z \xleftarrow{R} \{0,1\}^{\lambda+1} : B(z) = 1] \right|$

$= |p_i - p_{i+1}|$ since B outputs whatever A outputs

$= \varepsilon$

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

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

$$|p_0 - p_\ell| \leq |p_0 - p_1| + \cdots + |p_{\ell-1} - p_\ell|$$
$$\leq \ell \cdot \mathrm{negl}(\lambda)$$
$$= \mathrm{negl}(\lambda) \quad \text{since } \ell = \mathrm{poly}(\lambda). \blacksquare$$

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.

Pseudorandom functions (PRFs) — the "workhorse" of symmetric cryptography (will see applications next week)

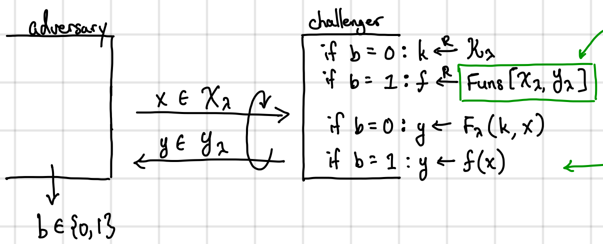
↳ PRG "compresses" a long random string into a short seed

↳ PRF "compresses" a random function into a short key

Definition. A pseudorandom function (PRF) with key-space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$, domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$, and range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ is a family of efficiently-computable functions $F = \{F_\lambda\}_{\lambda \in \mathbb{N}}$ where $F_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. We say that F is secure if for all efficient adversaries $\mathcal{A}$,

$$\mathrm{PRFAdv}[\mathcal{A}, F] = |\Pr[W_0 = 1] - \Pr[W_1 = 1]| = \mathrm{negl}(\lambda),$$

where for $b \in \{0,1\}$, we define W_b to be the output of $\mathcal{A}$ in the following experiment:



the set of all functions from $\mathcal{X}_\lambda \to \mathcal{Y}_\lambda$ (this set has size $|\mathcal{Y}_\lambda|^{|\mathcal{X}_\lambda|}$ – exponential size!)

for notational convenience, we will often drop the security parameter in the future (and leave it as implicitly defined)

Intuitively: A PRF is secure if no efficient adversary can distinguish the input/output behavior of the PRF (with a random and secret) key from the outputs of a truly random function (on the same domain and range).

↳ Very surprising: truly random functions do not have efficient representations, and yet we can mimic the behavior of such a function using a very short seed

↳ Will see many applications next week.

PRF ⇒ PRG. Suppose we have a PRF $F : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}$. We can construct a PRG $G : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ for any $\ell(\lambda) = \mathrm{poly}(\lambda)$ as follows:

$$G(s) := \underbrace{F(s,1) \,\|\, F(s,1) \,\|\, \cdots \,\|\, F(s,\ell)}_{\text{evaluate the PRF on inputs } 1 \text{ through } \ell}$$

(known as "counter mode")

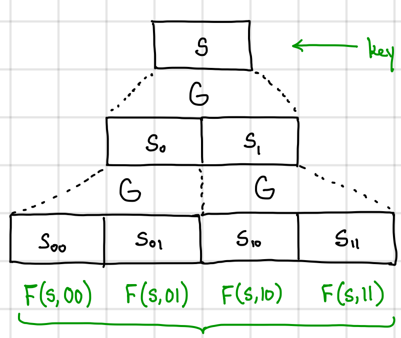
Security proof (sketch): When s is sampled uniformly at random, then $F(s, \cdot)$ is computationally indistinguishable from random function $f(\cdot)$. Now,

$$f(1) \,\|\, f(2) \,\|\, \cdots \,\|\, f(\ell)$$

is uniformly random string.

Formally, should set up a reduction.

PRG ⇒ PRF. (Goldreich-Goldwasser-Micali). Suppose we have a length-doubling PRG $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$.



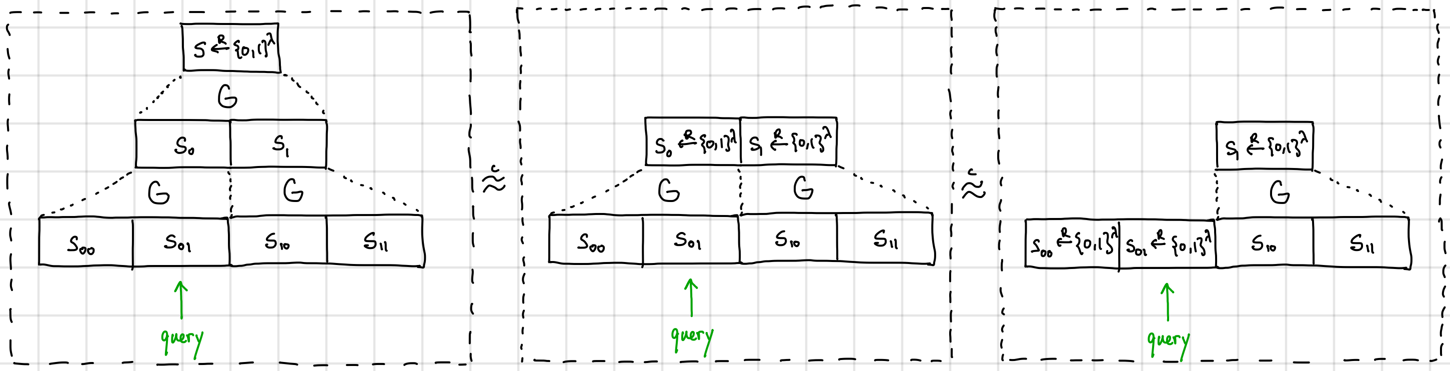
key for PRF

2-bit PRF → repeat this step n times to obtain a PRF on n-bit domains

Security proof: naïve approach is to replace all the leaf nodes with uniformly random values

exponentially many leaf nodes, so this requires exponentially-many hybrids, which is problematic

Alternative proof strategy is to proceed <u>query-by-query</u>: for each query, replace value of all intermediate nodes with random string



<u>Final hybrid</u>: adversary get random string for all of its queries

↳ Total number of hybrids is $Q \cdot n = \text{poly}(\lambda)$

PRF queries ↑ depth of tree

<u>Pseudorandom permutations (PRPs)</u>: Similar to PRFs, except replace the function with a <u>permutation</u>

<u>Definition</u>. A function family $F_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{X}_\lambda$ is a secure PRP if

1. For all keys $k \in \mathcal{K}_\lambda$, $F_\lambda(k, \cdot)$ is a permutation on $\mathcal{X}_\lambda$
2. The function family F_λ is pseudorandom ($F_\lambda(k,\cdot)$ is computationally indistinguishable from $f(\cdot)$ when $k \xleftarrow{R} \mathcal{K}_\lambda$ and $f \xleftarrow{R} \text{Perm}[\mathcal{X}_\lambda]$)

set of all permutations on $\mathcal{X}_\lambda$

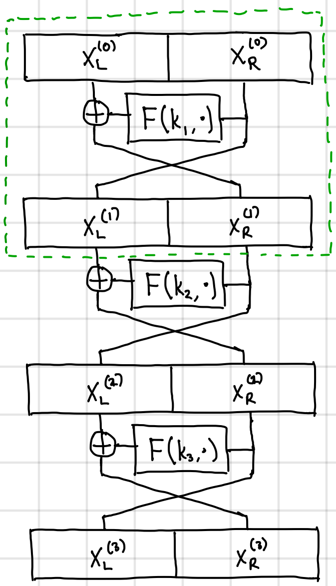
<u>PRP ⟹ PRF</u>. (PRF switching lemma). Suppose $F_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{X}_\lambda$ is a secure PRP. Then, for any efficient adversary making at most Q queries,

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

<u>Intuition</u>. If a PRP is over a large (i.e., super-polynomial) domain, then a PRP is as good as a PRF. Follows from fact that a random permutation behaves like a random function (up to the birthday bound - e.g., when a collision occurs)

<u>PRF ⟹ PRP</u>. (Luby-Rackoff). We can construct a PRP from any PRF using the 3-round <u>Feistel construction</u>:

Input: $x = x_L^{(0)} \| x_R^{(0)}$ (viewed as bit-strings)



Feistel round

Luby-Rackoff (Informal). If F is a secure PRF, then the 3-round Feistel network (shown to the left) is a secure PRP.

<u>Important</u>: independent "round" keys (k_1, k_2, k_3) used in the Feistel network

This construction is <u>invertible</u>.
(basic design of DES block cipher)

The story so far: Symmetric primitives: OWFs, PRGs, PRFs, PRPs

All of these primitives are closely related:

$$\begin{tikzcd}[column sep=huge]
\boxed{\text{OWFs}} \arrow[r, bend left, "\substack{\text{Goldreich-Levin, HILL} \\ \text{Blum-Micali}}"] & \text{PRGs} \arrow[r, bend left, "\text{Goldreich-Goldwasser-Micali}"] \arrow[l, bend left, "\text{immediate}"] & \text{PRFs} \arrow[r, bend left, "\text{Luby-Rackoff}"] \arrow[l, bend left, "\text{counter mode}"] & \text{PRPs} \arrow[l, bend left, "\text{switching lemma}"]
\end{tikzcd}$$

"simplest" notion that captures cryptographic hardness (often considered the fundamental primitive in symmetric cryptography)