From PRGs to PRFs

A two-dimensional hybrid argument

—Lecture 8—

Christopher Brzuska

March 4, 2024

1 Overview

Today, we show that pseudo-random generators (PRGs) imply pseudorandom functions (PRFs). In particular, we are going to cover

(1) how to construct a PRF from a length-doubling pseudorandom generator (PRG) via the Goldreich-Goldwasser-Micali (GGM) construction.

(2) two-dimensional telescopic arguments

(3) reductions that depend on the number of adversarial queries

On Exercise Sheet 7, we prepare for the Goldreich-Levin hardcore bit proof which we will cover next week. In order to prepare for this, in the very end of this lecture, we also cover

(4) averaging arguments (Markov)

(5) tail bounds (Chernoff)

In addition, the Exercise Sheet 7 will cover a simplified version of the Goldreich-Levin proof.

$\underline{\mathsf{Gprg}^0_G}$

Parameters

λ:	security par.
$s(\lambda)$:	length-exp.
G:	function

State

y:	image value

SAMPLE()

assert $y = \bot$

$x \leftarrow\!\!\$\ \{0,1\}^\lambda$

$y \leftarrow G(x)$

return y

$\underline{\mathsf{Gprg}^1_s}$

Parameters

λ:	security par.
$s(\lambda)$:	length-exp.

State

y:	random value

SAMPLE()

assert $y = \bot$

$y \leftarrow\!\!\$\ \{0,1\}^{\lambda+s(\lambda)}$

return y

Figure 1: SAMPLE oracles of the games $\mathtt{Gprg}^0_G$ and $\mathtt{Gprg}^1_s$

2 Definitions

We recall that a pseudorandom function (PRF) is a function that is indistinguishable from a truly random function. The definition is given in Figure 3. Recall that we use the notation $\mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{Gprf}_f^0$ for denoting that the adversary interacts with the EVAL oracle of the real PRF game $\mathtt{Gprf}_f^0$ and the notation $\mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{Gprf}^1_{\mathrm{keyl,in,out}}$ for denoting that the adversary interacts with the EVAL oracle of the ideal PRF game $\mathtt{Gprf}^1_{\mathrm{keyl,in,out}}$. In these expressions, the adversary can call the EVAL oracle multiple times, and a secure pseudorandom function should have the property that the adversary cannot distinguish the oracle outputs produced by the real pseudorandom function from the oracle outputs produced by an ideal random function. As before, we capture distinguishing by measuring how likely it is that adversary returns 1 in either of the settings. Recall that we denote by

$\mathtt{Gprf}_f^0$

Package Parameters	
λ:	security parameter
keyl:	key length
in:	input length
out:	output length
f:	(keyl, in, out)-PRF

Package State	
k:	*key*

EVAL(x)

assert $x \in \{0,1\}^{\mathrm{in}(\lambda)}$
if $k = \bot$:
  $k \leftarrow\$ \{0,1\}^{\mathrm{keyl}(\lambda)}$
$y \leftarrow f(k, x)$
return y

$\mathtt{Gprf}^1_{\mathrm{keyl,in,out}}$

Package Parameters	
λ:	security param.
keyl:	key length
in:	input length
out:	output length

Package State	
T:	**table** [bitstring → bitstring]

EVAL(x)

assert $x \in \{0,1\}^{\mathrm{in}(\lambda)}$
if $T[x] = \bot$
  $T[x] \leftarrow\$ \{0,1\}^{\mathrm{out}(\lambda)}$
$y \leftarrow T[x]$
return y

Figure 2: EVAL(x) oracles of the ideal $\mathtt{Gprf}^1_{\mathrm{keyl,in,out}}$ and the real $\mathtt{Gprf}^0_G$ games

$$\Pr\left[1 = \mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{Gprf}_f^0\right]$$

the probability that $\mathcal{A}$ returns 1 when interacting with the EVAL oracle of the $\mathtt{Gprf}_f^0$ game. The probability here is over drawing k uniformly at random from $\{0,1\}^\lambda$ as well as the adversary's own randomness. I.e., probability denotes the fraction of these strings that yields answer 1 from the adversary. We denote by

$$\Pr\left[1 = \mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{Gprf}^1_{\mathrm{keyl,in,out}}\right]$$

the probability that $\mathcal{A}$ returns 1 when interacting with the EVAL oracle of the $\mathtt{Gprf}^1_{\mathrm{keyl,in,out}}$ game. The probability here is over drawing the answers in EVAL uniformly at random each time that the oracle is called as well as the adversary's own randomness. Indistinguishability sais that these two probability should be roughly equal. Recall that each algorithm gets the security parameter implicitly.

Definition 2.1 (Pseudorandom Function)**.** A deterministic, polynomially-time computable function $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ is a pseudorandom function with keyl, in, and out if it satisfies the following:

- **Length-condition:** For all $\lambda \in \mathbb{N}$, for all $k \in \{0,1\}^{\text{keyl}(\lambda)}$, $x \in \{0,1\}^{\text{in}(\lambda)}$, $|f(k,x)| = \text{out}(\lambda)$.
- **Pseudorandomness:** For all PPT $\mathcal{A}$,

$$\mathbf{Adv}^{\mathcal{A},f}_{\mathsf{PRF}}(\lambda) := \left|\Pr\left[1 = \mathcal{A}^{\ \mathsf{EVAL}\ \to}\ \mathtt{Gprf}^0_f\right] - \Pr\left[1 = \mathcal{A}^{\ \mathsf{EVAL}\ \to}\ \mathtt{Gprf}^1_{\text{keyl,in,out}}\right]\right|$$

  is negligible in λ.

Remark Throughout this lecture, we use $\text{keyl}(\lambda) = \text{in}(\lambda) = \text{out}(\lambda) = \lambda$. Recall that a pseudorandom generator (PRG) is a *deterministic* function that takes a truly random bitstring and turns it into a longer, *pseudorandom* bitstring. Today, we only need a length-doubling PRG G, i.e, $s(\lambda) = 2\lambda$. As last week, we use real game $\mathtt{Gprg}^0_G$ and ideal game $\mathtt{Gprg}^1_{s(\lambda)=\lambda}$ which both expose an oracle SAMPLE to model PRG security of G and ask that for all PPT adversaries $\mathcal{A}$, the difference

$$\mathsf{Adv}^{\mathsf{PRG}}_{G,\mathcal{A}}(\lambda) := \left|\Pr\left[1 = \mathcal{A}^{\ \mathsf{SAMPLE}\ \to}\ \mathtt{Gprg}^0_G\right] - \Pr\left[1 = \mathcal{A}^{\ \mathsf{SAMPLE}\ \to}\ \mathtt{Gprg}^1_{s(\lambda)=\lambda}\right]\right|$$

is negligible in λ.

3 The Goldreich-Goldwasser-Micali (GGM) construction

The Goldreich-Goldwasser-Micali (GGM) construction constructs a pseudorandom function from a length-doubling pseudorandom generator as illustrated in Figure 3.

Recall the intuition of the GGM construction. Intuitively, a PRF needs to generate a pseudorandom value for each $x \in \{0,1\}^\lambda$. The GGM construction does so by building a binary tree of depth $|x|$. On level 0, there is only $2^0 = 1$ value, namely the key k. On level 1, there are then $2^1 = 2$ values, namely $G_0(k)$ and $G_1(k)$. On level 2, there are $2^2 = 4$ values, namely[1] $G_0 \circ G_0(k)$, $G_1 \circ G_0(k)$, $G_0 \circ G_1(k)$ and $G_1 \circ G_1(k)$. On level 3, there are $2^3 = 8$ values and on level n, there are the desired number 2^λ value. However, storing all 2^λ values takes exponential space. Thus, instead of storing all these values, the function f just computes them on-the-fly when evaluated on an input $x \in \{0,1\}^\lambda$. The input x here then represents the address in the binary tree. This way, each evaluation of f only corresponds to $|x| = \lambda$ evaluations of the PRG G.

$f(k,x)$

$s_0 \leftarrow k$

for i **from** 1 **to** $|x|$:

  $s_i \leftarrow G_{x[i]}(s_0)$

return $s_{|x|}$

Figure 4: GGM constructs PRF f from length-doubling PRG $G = G_0||G_1$

Theorem (Goldreich-Goldwasser-Micali (GGM))**.** Let G be a pseudorandom generator (PRG) with $|G(s)| = 2|x|$. We write $G(s) = G_0(s)||G_1(s)$, where

[1] We here use the $\circ$ notation so we can avoid brackets, i.e., we write $G_0 \circ G_1(k)$ instead of $G_0(G_1(x))$

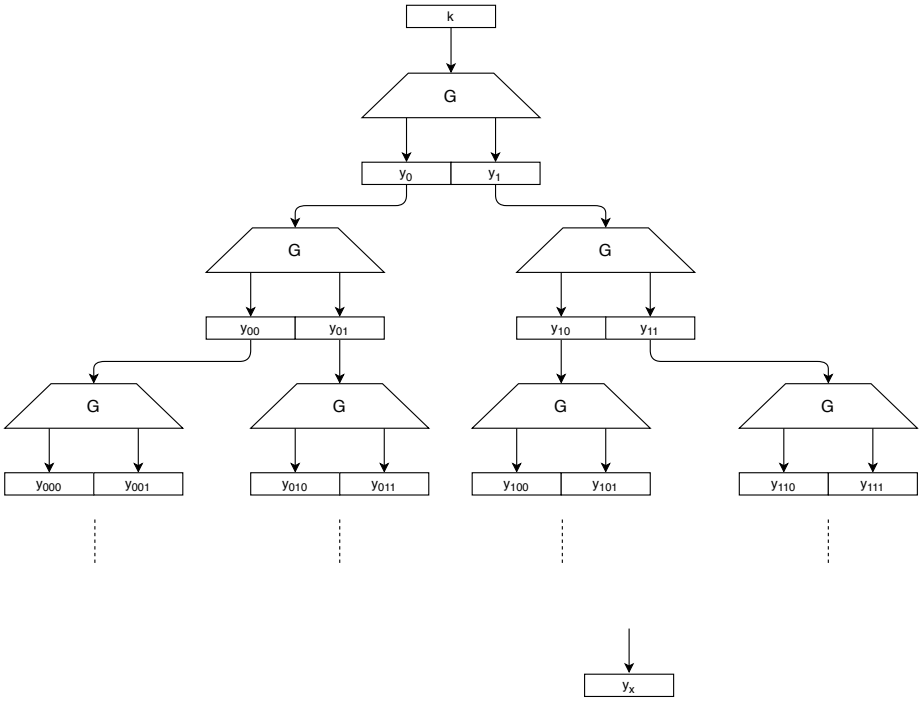

Figure 3: The GGM construction of a PRF $f(k, \cdot)$ based on a length-doubling PRG G. Here, the output $f(k, x)$ is marked as y_x. The path through the tree corresponds to the bits of the *input* x of the PRF, whereas the *key of the PRF* becomes the *input of the PRG*. This is important, because PRG security only holds when the PRG's input is uniformly random (or pseudorandom). The key k is uniformly random, whereas the input x is not.

$|G_0(s)| = |G_1(s)| = |x|$. Then, f is a pseudorandom function, where f is defined in Figure 4. In particular, for every polynomial $q(\lambda)$, there is a PPT reduction $\mathcal{R}$ such that for all PPT adversaries $\mathcal{A}$ that make at most $q(\lambda)$ queries, it holds that

$$\mathsf{Adv}^{\mathsf{PRF}}_{f,\mathcal{A}}(\lambda) \leq \lambda \cdot q(\lambda) \cdot \mathsf{Adv}^{\mathsf{PRG}}_{G,\mathcal{A}\to\mathcal{R}}(\lambda),$$

i.e.,

$$\left|\Pr\left[1 = \mathcal{A} \overset{\mathsf{EVAL}}{\to} \mathtt{Gprf}^0_f\right] - \Pr\left[1 = \mathcal{A} \overset{\mathsf{EVAL}}{\to} \mathtt{Gprf}^1_{\lambda,\lambda,\lambda}\right]\right|$$
$$= (q(\lambda))^2 \cdot \left|\Pr\left[1 = \mathcal{A} \overset{\mathsf{EVAL}}{\to} \mathcal{R} \overset{\mathsf{SAMPLE}}{\to} \mathtt{Gprg}^0_G\right] - \Pr\left[1 = \mathcal{A} \overset{\mathsf{EVAL}}{\to} \mathcal{R} \overset{\mathsf{SAMPLE}}{\to} \mathtt{Gprg}^1_{s(\lambda)=\lambda}\right]\right|$$

For the theorem to be meaningful, it would suffice to prove $\neq$, but we actually prove equality. Note that the reduction $\mathcal{R}$ needs to "know" $q(\lambda)$, the number of queries made by $\mathcal{A}$, or at least an upper bound on that number. We now turn to the proof of Theorem 3. Similar to last week, the idea is to have a *hybrid* argument which slowly moves from outputs of the real PRF f to outputs from a uniformly random function. Interestingly, this time, we carry out a hybrid over the *depth* of the GGM tree *and* over the *number of queries* made by the adversaries. I.e., the hybrid games are indexed by a pair (d, j). The idea of the

$\underline{\underline{\mathcal{H}_{d,j}}}$	$\underline{\underline{\mathcal{R}}}$	$\underline{\underline{\mathcal{R}_{d,j}}}$
$\mathsf{EVAL}(x)$	$\mathsf{EVAL}(x)$	$\mathsf{EVAL}(x)$
assert $\|x\| = \lambda$	**assert** $\|x\| = \lambda$	**assert** $\|x\| = \lambda$
if $c = \perp$ **then**	**if** $c = \perp$ **then**	**if** $c = \perp$ **then**
$c \leftarrow 0$	$d \leftarrow\!\!\$\ \{1, .., \lambda\}$	
	$j \leftarrow\!\!\$\ \{1, .., q(\lambda)\}$	
	$c \leftarrow 0$	$c \leftarrow 0$
if $T[x[1..d\text{-}1]] = \perp$ **then**	**if** $T[x[1..d\text{-}1]] = \perp$ **then**	**if** $T[x[1..d\text{-}1]] = \perp$ **then**
$c \leftarrow c + 1$	$c \leftarrow c + 1$	$c \leftarrow c + 1$
if $c \leq j$ **then**	**if** $c < j$ **then**	**if** $c < j$ **then**
$y_0 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$y_0 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$y_0 \leftarrow\!\!\$\ \{0,1\}^\lambda$
$y_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$y_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$y_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$
	if $c = j$ **then**	**if** $c = j$ **then**
	$y_0 \| y_1 \leftarrow \mathsf{SAMPLE}$	$y_0 \| y_1 \leftarrow \mathsf{SAMPLE}$
if $c > j$ **then**	**if** $c > j$ **then**	**if** $c > j$ **then**
$s \leftarrow\!\!\$\ \{0,1\}^\lambda$	$s \leftarrow\!\!\$\ \{0,1\}^\lambda$	$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
$y_0 \leftarrow G_0(s)$	$y_0 \leftarrow G_0(s)$	$y_0 \leftarrow G_0(s)$
$y_1 \leftarrow G_1(s)$	$y_1 \leftarrow G_1(s)$	$y_1 \leftarrow G_1(s)$
$T[x[1..d\text{-}1]](0) \leftarrow y_0$	$T[x[1..d\text{-}1]](0) \leftarrow y_0$	$T[x[1..d\text{-}1]](0) \leftarrow y_0$
$T[x[1..d\text{-}1]](1) \leftarrow y_1$	$T[x[1..d\text{-}1]](1) \leftarrow y_1$	$T[x[1..d\text{-}1]](1) \leftarrow y_1$
$s_d \leftarrow T[x[1..d\text{-}1]](x[d])$	$s_d \leftarrow T[x[1..d\text{-}1]](x[d])$	$s_d \leftarrow T[x[1..d\text{-}1]](x[d])$
for i **from** $d+1$ **to** $\|x\|$:	**for** i **from** $d+1$ **to** $\|x\|$:	**for** i **from** $d+1$ **to** $\|x\|$:
$s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$	$s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$	$s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$
return s_λ	**return** s_λ	**return** s_λ

Figure 5: Hybrid game $\mathtt{H}_{d,j}$, reduction $\mathcal{R}$ and reduction $\mathcal{R}_{d,j}$

reduction is to work level by level, i.e., first replace the root layer application of the PRG by a uniformly random string, then replace the PRG application on the second layer by a random string and so on. We do not replace all 2^i applications of the PRG by a uniformly random string, but rather, we only replace the application *at the points where the adversary makes a query* since this is some polynomial $q(\lambda)$ rather than some exponential value. Therefore, we have that

$$1 \leq d \leq \lambda$$
$$0 \leq j \leq q(\lambda)$$

We now make this proof idea formal. Let $\mathcal{A}$ be a PPT adversary against the PRF f which makes at most $q(\lambda)$ queries to the EVAL oracle. Then, for $1 \leq d \leq \lambda$ and $0 \leq j \leq q(\lambda)$, we define the game $\mathtt{H}_{d,j}$ as in Figure 5. We now need to show the following three claims:

$\underline{\underline{\mathtt{Gprf}^0_f}}$	$\underline{\underline{\mathtt{Gprf}^1_{\lambda,\lambda,\lambda}}}$
$\underline{\mathsf{EVAL}(x)}$	$\underline{\mathsf{EVAL}(x)}$
assert $x \in \{0,1\}^\lambda$	**assert** $x \in \{0,1\}^\lambda$
if $k = \bot$:	**if** $T[x] = \bot$
$\quad k \leftarrow\!\$\ \{0,1\}^\lambda$	$\quad T[x] \leftarrow\!\$\ \{0,1\}^\lambda$
$y \leftarrow f(k,x)$	$y \leftarrow T[x]$
return y	**return** y

Figure 6: $\mathsf{EVAL}(x)$ oracles of the ideal $\mathtt{Gprf}^1_{\lambda,\lambda,\lambda}$ and the real $\mathtt{Gprf}^0_f$ games.

Claim 1 (Extreme Hybrids)**.**

$$\mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{H}_{1,0} \overset{\text{code}}{\equiv} \mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{Gprf}^0_f$$
$$\mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{H}_{\lambda,q(\lambda)} \overset{\text{code}}{\equiv} \mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{Gprf}^1_{\lambda,\lambda,\lambda}$$

Claim 2 (Compatible Hybrids across layers)**.** For all $1 \leq d \leq \lambda$:

$$\mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{H}_{d-1,q(\lambda)} \overset{\text{code}}{\equiv} \mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{H}_{d,0}$$

Note that both, Claim 1 and Claim 2 include the adversary into the statement, since it is important that the adversary makes at most $q(\lambda)$ queries. Else, the two games would actually not be equivalent. The third claim relates to the existence of a reduction for steps *within a single layer*.

Claim 3 (Reduction)**.** For all $1 \leq d \leq \lambda$, $1 \leq j \leq q(\lambda)$:

$$\mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathcal{R}_{d,j} \overset{\mathsf{SAMPLE}}{\rightarrow} \mathtt{Gprg}^0_G \overset{\text{code}}{\equiv} \mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{H}_{d,j\text{-}1} \tag{1}$$
$$\mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathcal{R}_{d,j} \overset{\mathsf{SAMPLE}}{\rightarrow} \mathtt{Gprg}^1_\lambda \overset{\text{code}}{\equiv} \mathcal{A} \overset{\mathsf{EVAL}}{\rightarrow} \mathtt{H}_{d,j}, \tag{2}$$

where $\mathcal{R}_{d,j}$ is defined in Figure 5.

We first state them, then show why they suffice (using the telescopic sum argument over pairs) and then prove each of the claims.
Claim 1 states that the *extreme* variants of the hybrid game $\mathtt{H}_{1,0}$ and $\mathtt{H}_{\lambda,q(\lambda)}$ are equivalent to $\mathtt{Gprf}^0_f$ and $\mathtt{Gprf}^1_{\lambda,\lambda,\lambda}$, respectively—as long as the adversary $\mathcal{A}$ makes at most q queries. In the proof of the equivalence between $\mathtt{Gprf}^1_{\lambda,\lambda,\lambda}$ and $\mathtt{Gprf}^1_{\lambda,\lambda,\lambda}$, we rely on the number of queries.

Claim 2 states that the last hybrid game of layer d-1 is code-equivalent to the first hybrid game of layer d. In this equivalence proof, we again rely on the number of queries.
Finally, Claim 3 shows that that we can use the reduction $\mathcal{R}_{d,j}$ to show that two subsequent hybrids are indistinguishable.
We now show why Claim 1, Claim 2 and Claim 3 suffice to prove Theorem 3. The first equality follows by Claim 1. The second equality is a telescopic sum over the layers, i.e., due to Claim 2 all terms except for the extreme terms cancel out, making the telescopic sum equal to the previous line. Similarly, the third equality is a telescopic sum within each layer and thus, per layer, all except for the extreme hybrids cancel out. Then, we rely on Claim 3 to re-write $\mathtt{H}_{d,j}$ in terms of the reduction. Then, we split the two sums and multiply by $1 = \frac{\lambda \cdot q(\lambda)}{\lambda \cdot q(\lambda)}$. Finally, for the last equality, we observe that the probability over drawing $d \leftarrow\!\!\$\ \{1,..,\lambda\}$ and $j \leftarrow\!\!\$\ \{1,..,q(\lambda)\}$ that $\mathcal{R} = \mathcal{R}_{d,j}$ is $\frac{1}{s(\lambda)}$. In fact, verifying this equality works best when thinking about it in the inverse direction. I.e., start with $\mathcal{R}$ and then observer that $\mathcal{R}$ draws a specific d with probability $\frac{1}{\lambda}$ and a specific j with probability $\frac{1}{q(\lambda)}$.

$$
\begin{aligned}
&\left|\Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathtt{Gprf}_f^0\right] - \Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathtt{Gprf}_{\lambda,\lambda,\lambda}^1\right]\right| \\
=&\left|\Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathtt{H}_{1,0}\right] - \Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathtt{H}_{\lambda,q(\lambda)}\right]\right| \text{ (Claim 1)} \\
=&\left|\sum_{1\le d\le\lambda} \Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathtt{H}_{d,0}\right] - \Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathtt{H}_{d,q(\lambda)}\right]\right| \text{ (Telescope I + Claim 2)} \\
=&\left|\sum_{1\le d\le\lambda,1\le j\le q(\lambda)} \Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathtt{H}_{d,j\text{-}1}\right] - \Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathtt{H}_{d,j}\right]\right| \text{ (Telescope II)} \\
=&\left|\sum_{1\le d\le\lambda,1\le j\le q(\lambda)} \Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathcal{R}_{d,j}^{\,\mathsf{SAMPLE}}\to \mathtt{Gprg}_G^0\right] - \Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathcal{R}_{d,j}^{\,\mathsf{SAMPLE}}\to \mathtt{Gprg}_\lambda^1\right]\right| \text{ (Claim 3)} \\
=&\Big|\lambda\cdot q(\lambda)\cdot\left(\sum_{1\le d\le\lambda,1\le j\le q(\lambda)} \tfrac{1}{\lambda\cdot q(\lambda)}\cdot\Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathcal{R}_{d,j}^{\,\mathsf{SAMPLE}}\to \mathtt{Gprg}_G^0\right]\right) \\
&-\lambda\cdot q(\lambda)\left(\sum_{1\le d\le\lambda,1\le j\le q(\lambda)} \tfrac{1}{\lambda\cdot q(\lambda)}\cdot\Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathcal{R}_{d,j}^{\,\mathsf{SAMPLE}}\to \mathtt{Gprg}_\lambda^1\right]\right)\Big| \\
=&\lambda\cdot q(\lambda)\cdot\left|\Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathcal{R}^{\,\mathsf{SAMPLE}}\to \mathtt{Gprg}_G^0\right] - \Pr\left[1 = \mathcal{A}^{\,\mathsf{EVAL}}\to \mathcal{R}^{\,\mathsf{SAMPLE}}\to \mathtt{Gprg}_\lambda^1\right]\right|
\end{aligned}
$$

We can prove Claim 1 via code comparison and omit it from these lecture notes. It is analogous to the code comparison for the extreme hybrids that we considered last week.
We prove Claim 3 via inlining on page 8 and page 9. The left column contains the code of $\mathcal{R}_{d,j}$. The second column contains the code of $\mathcal{R}_{d,j+1}$ with $\mathtt{Gprg}_G^1$. From column 2 to column 3, we then merge the if-condition for $c < j$ and $c = j$, since the both sample y_0 and y_1 each independently and uniformly at random from $\{0,1\}^\lambda$. The code of column 3 is the code of $\mathtt{H}_{d,j}$ which concludes the proof

of the first equation of Claim 2. For the second equation of Claim 2, see page 9. This time, we merge the if-condition $c = j$-1 and $c > j$-1.
We sketch the proof of Claim 2 by comparing $\mathtt{H}_{d-1,j}$ and $\mathtt{H}_{d,0}$ (see page 9). In the last column, $j = 0$ and thus, it is easy to see that $c < 0$ never occurs and thus, the first **for** loop can be omitted. In turn, in the first column, $j = q(\lambda)$ and we rely on the adversary not making more than $q(\lambda)$ queries to ensure that the case $c > j = q(\lambda)$ never happens and thus, the second **for** loop can be omitted. We can now join the **for** loop and the case $c = j$ and thus can remove all assignments and operations on the counter c. While their code is not identical, their input-output behaviour of the resulting code is. We omit the details.

$\underline{\underline{\mathcal{R}_{d,j}}}$
$\mathsf{EVAL}(x)$

assert $|x| = \lambda$
if $c = \perp$ **then**
  $c \leftarrow 0$
if $T[x[1..d\text{-}1]] = \perp$ **then**
  $c \leftarrow c + 1$
  if $c < j$ **then**
    $y_0 \leftarrow\$ \{0,1\}^\lambda$
    $y_1 \leftarrow\$ \{0,1\}^\lambda$
  if $c = j$ **then**
    $y_0 || y_1 \leftarrow \mathsf{SAMPLE}$
  if $c > j$ **then**
    $s \leftarrow\$ \{0,1\}^\lambda$
    $y_0 \leftarrow G_0(s)$
    $y_1 \leftarrow G_1(s)$
  $T[x[1..d\text{-}1]](0) \leftarrow y_0$
  $T[x[1..d\text{-}1]](1) \leftarrow y_1$
$s_d \leftarrow T[x[1..d\text{-}1]](x[d])$
for i **from** $d+1$ **to** λ :
  $s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$
return s_λ

$\underline{\underline{\mathcal{R}_{d,j}^{\ \mathsf{SAMPLE}} \mathtt{Gprg}_\lambda^1}}$
$\mathsf{EVAL}(x)$

assert $|x| = \lambda$
if $c = \perp$ **then**
  $c \leftarrow 0$
if $T[x[1..d\text{-}1]] = \perp$ **then**
  $c \leftarrow c + 1$
  if $c < j$ **then**
    $y_0 \leftarrow\$ \{0,1\}^\lambda$
    $y_1 \leftarrow\$ \{0,1\}^\lambda$
  if $c = j$ **then**
    $y_0 || y_1 \leftarrow\$ \{0,1\}^{2\lambda}$
  if $c > j$ **then**
    $s \leftarrow\$ \{0,1\}^\lambda$
    $y_0 \leftarrow G_0(s)$
    $y_1 \leftarrow G_1(s)$
  $T[x[1..d\text{-}1]](0) \leftarrow y_0$
  $T[x[1..d\text{-}1]](1) \leftarrow y_1$
$s_d \leftarrow T[x[1..d\text{-}1]](x[d])$
for i **from** $d+1$ **to** λ :
  $s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$
return s_λ

$\underline{\underline{\mathcal{H}_{d,j}}}$
$\mathsf{EVAL}(x)$

assert $|x| = \lambda$
if $c = \perp$ **then**
  $c \leftarrow 0$
if $T[x[1..d\text{-}1]] = \perp$ **then**
  $c \leftarrow c + 1$
  if $c \leq j$ **then**
    $y_0 \leftarrow\$ \{0,1\}^\lambda$
    $y_1 \leftarrow\$ \{0,1\}^\lambda$
  if $c > j$ **then**
    $s \leftarrow\$ \{0,1\}^\lambda$
    $y_0 \leftarrow G_0(s)$
    $y_1 \leftarrow G_1(s)$
  $T[x[1..d\text{-}1]](0) \leftarrow y_0$
  $T[x[1..d\text{-}1]](1) \leftarrow y_1$
$s_d \leftarrow T[x[1..d\text{-}1]](x[d])$
for i **from** $d+1$ **to** λ :
  $s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$
return s_λ

$\underline{\underline{\mathcal{R}_{d,j}}}$
$\underline{\mathsf{EVAL}(x)}$

assert $|x| = \lambda$
if $c = \perp$ **then**
  $c \leftarrow 0$
if $T[x[1..d\text{-}1]] = \perp$ **then**
  $c \leftarrow c + 1$
  if $c < j$ **then**
    $y_0 \leftarrow\$ \{0,1\}^\lambda$
    $y_1 \leftarrow\$ \{0,1\}^\lambda$
  if $c = j$ **then**
    $y_0 || y_1 \leftarrow \mathsf{SAMPLE}$
  if $c > j$ **then**
    $s \leftarrow\$ \{0,1\}^\lambda$
    $y_0 \leftarrow G_0(s)$
    $y_1 \leftarrow G_1(s)$
  $T[x[1..d\text{-}1]](0) \leftarrow y_0$
  $T[x[1..d\text{-}1]](1) \leftarrow y_1$
$s_d \leftarrow T[x[1..d\text{-}1]](x[d])$
for i **from** $d+1$ **to** λ :
  $s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$
return s_λ

$\underline{\underline{\mathcal{R}_{d,j} \overset{\mathsf{SAMPLE}}{\rightarrow} \mathtt{Gprg}_G^0}}$
$\underline{\mathsf{EVAL}(x)}$

assert $|x| = \lambda$
if $c = \perp$ **then**
  $c \leftarrow 0$
if $T[x[1..d\text{-}1]] = \perp$ **then**
  $c \leftarrow c + 1$
  if $c < j$ **then**
    $y_0 \leftarrow\$ \{0,1\}^\lambda$
    $y_1 \leftarrow\$ \{0,1\}^\lambda$
  if $c = j$ **then**
    $s \leftarrow\$ \{0,1\}^\lambda$
    $y_0 || y_1 \leftarrow G(s)$
  if $c > j$ **then**
    $s \leftarrow\$ \{0,1\}^\lambda$
    $y_0 \leftarrow G_0(s)$
    $y_1 \leftarrow G_1(s)$
  $T[x[1..d\text{-}1]](0) \leftarrow y_0$
  $T[x[1..d\text{-}1]](1) \leftarrow y_1$
$s_d \leftarrow T[x[1..d\text{-}1]](x[d])$
for i **from** $d+1$ **to** λ :
  $s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$
return s_λ

$\underline{\underline{\mathcal{H}_{d,j\text{-}1}}}$
$\underline{\mathsf{EVAL}(x)}$

assert $|x| = \lambda$
if $c = \perp$ **then**
  $c \leftarrow 0$
if $T[x[1..d\text{-}1]] = \perp$ **then**
  $c \leftarrow c + 1$
  if $c \leq j\text{-}1$ **then**
    $y_0 \leftarrow\$ \{0,1\}^\lambda$
    $y_1 \leftarrow\$ \{0,1\}^\lambda$
  if $c > j\text{-}1$ **then**
    $s \leftarrow\$ \{0,1\}^\lambda$
    $y_0 \leftarrow G_0(s)$
    $y_1 \leftarrow G_1(s)$
  $T[x[1..d\text{-}1]](0) \leftarrow y_0$
  $T[x[1..d\text{-}1]](1) \leftarrow y_1$
$s_d \leftarrow T[x[1..d\text{-}1]](x[d])$
for i **from** $d+1$ **to** λ :
  $s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$
return s_λ

$\underline{\mathcal{H}_{d\text{-}1,q(\lambda)}}$ $\underline{\mathsf{EVAL}(x)}$	$\underline{\mathcal{H}_{d,0}}$ $\underline{\mathsf{EVAL}(x)}$
assert $\lvert x\rvert = \lambda$	**assert** $\lvert x\rvert = \lambda$
if $c = \perp$ **then**	**if** $c = \perp$ **then**
$c \leftarrow 0$	$c \leftarrow 0$
if $T[x[1..d\text{-}2]] = \perp$ **then**	**if** $T[x[1..d\text{-}1]] = \perp$ **then**
$c \leftarrow c + 1$	$c \leftarrow c + 1$
if $c \leq q(\lambda)$ **then**	**if** $c \leq 0$ **then**
$y_0 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$y_0 \leftarrow\!\!\$\ \{0,1\}^\lambda$
$y_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$	$y_1 \leftarrow\!\!\$\ \{0,1\}^\lambda$
if $c > q(\lambda)$ **then**	**if** $c > 0$ **then**
$s \leftarrow\!\!\$\ \{0,1\}^\lambda$	$s \leftarrow\!\!\$\ \{0,1\}^\lambda$
$y_0 \leftarrow G_0(s)$	$y_0 \leftarrow G_0(s)$
$y_1 \leftarrow G_1(s)$	$y_1 \leftarrow G_1(s)$
$T[x[1..d\text{-}2]](0) \leftarrow y_0$	$T[x[1..d\text{-}1]](0) \leftarrow y_0$
$T[x[1..d\text{-}2]](1) \leftarrow y_1$	$T[x[1..d\text{-}1]](1) \leftarrow y_1$
$s_{d\text{-}11} \leftarrow T[x[1..d\text{-}2]](x[d\text{-}11])$	$s_d \leftarrow T[x[1..d\text{-}1]](x[d])$
for i **from** d **to** λ :	**for** i **from** $d+1$ **to** λ :
$s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$	$s_i \leftarrow G_{x[i]}(s_{i\text{-}1})$
return s_λ	**return** s_λ

4 Markov and Chernoff Bound

Next week, we want to perform some statistical analysis of algorithms, and we will need several statistical tools in order to do this. For an algorithm $\mathcal{A}$ that takes λ random coins r and returns a natural number or a real number, we can define the *expectation* of $\mathcal{A}$ as

$$\mathbb{E}(\mathcal{A}) := \sum_{r \in \{0,1\}^\lambda} 2^{-\lambda} \mathcal{A}(r),$$

which is the average value that $\mathcal{A}$ returns when taking a uniformly random r as input. We have seen such probability statements already many times in the course in the case where the algorithm (adversary) $\mathcal{A}$ returns 0 or 1. Note that each r is chosen with probability $2^{-\lambda}$. We call an algorithm that maps random strings to a real number a *random variable*. We will now see two bounds that tell us whether a random variable is likely to be far from its expectation, and if so, how likely. To appreciate the quality of the different bounds, let us consider the following example:

> Example: We flip a coin which has a 0 on one side and a 1 on the other side, such that the probability of getting 0 is $\frac{1}{2}$. Now, if we

> flip a coin 1000 times, we get 500 zeroes in expectation. How likely is it that we get much more than this, say, that we get 750 zeroes?

My intuition says that this should be quite unlikely. Let's now look at some popular tail bounds and see what they say about our example. The first bound is the Markov bound that is valid for all random variables that are positive. In the lemma, we replace the explicit notation $\Pr_{r \leftarrow \$\{0,1\}^n}[\mathcal{A}(r) \geq v]$ by the implicit notation $\Pr[\mathcal{A} \geq v]$, since this allows us to speak about arbitrary randomized algorithms/random variables without making explicit how long the random string is.

Lemma 1 (Markov Bound)**.** For all non-negative random variables $\mathcal{A}$ and all positive real numbers v, we have

$$\Pr[\mathcal{A} \geq v] \leq \frac{\mathbb{E}(\mathcal{A})}{v}$$

and

$$\Pr[\mathcal{A} \geq v \cdot \mathbb{E}(\mathcal{A})] \leq \frac{1}{v}.$$

The expectation of a random variable gives some information about a random variable, but not necessarily very much. For instance, the Markov bound only tells us some very weak relation between a random variable and its expectation. The Markov bound is not particularly good for repeated, independent experiments such as our example. However, Markov bound only tells us that this probability is lower than $\frac{2}{3}$ (which is not very informative). Therefore, we need better bounds such as the Chernoff bound (below) which tells us that the probability is really small. We now turn to repeated experiments with $0-1$ random variables Exp, i.e, random variables that either return 0 or 1 such as our security experiments.

Lemma 2 (Chernoff Bound)**.** For all $p \leq \frac{1}{2}$, for all $\mathsf{Exp}_1, \mathsf{Exp}_2, ..., \mathsf{Exp}_n$ independent 0-1 random variables so that for all i $\Pr[\mathsf{Exp}_i = 1] = p$, for all ϵ, $0 < \epsilon \leq p(1-p)$, we have

$$\Pr\left[\left|\frac{\sum_{i=1}^{n} \mathsf{Exp}_i}{n} - p\right| > \epsilon\right] < 2 \cdot e^{-\epsilon^2 n}$$

and

$$\Pr\left[\sum_{i=1}^{n} \mathsf{Exp}_i > (p+\epsilon)n\right] < 2 \cdot e^{-\epsilon^2 n}$$

In our example, $p = \frac{1}{2}$, $n = 1000$ and $\epsilon = \frac{1}{4}$, which yields that the probability of obtaining more than 750 zeroes is very small.