PRGs: From 1 bit to polynomially many

An example of the hybrid argument technique

—Lecture 7—

Christopher Brzuska

February 28, 2022

1 Teaching Period IV: Hardness amplification in cryptography

In the end of teaching period III, we learned that *distributional* one-wayness[1] can be amplified to *weak* one-wayness (using information-theoretic extractors), and weak one-wayness can be amplified to standard one-wayness via parallel repetition (or complexity-theoretic hash-functions). These were our first amplification results. From the introductory course in cryptography, we know that moreover, one-way functions (OWFs) imply pseudorandom generators (PRGs) and that PRGs imply pseudorandom functions (PRFs). In addition, PRFs also imply OWFs, so all three primitives are equivalent.

$$\exists\text{OWFs} \Leftrightarrow \exists\text{PRGs} \Leftrightarrow \exists\text{PRFs}$$

While we gave some intuition for why this is true, we also discussed that OWFs are a very very very weak notion of security (although in Lecture 5, we saw even weaker notions...). So, how can it be that we can build something as strong as a cipher/PRF just from a OWF? Understanding how this exactly works is the goal of this teaching period as well as deepening our understanding of the limits of what we can prove. Below is a (preliminary) outline of the contents:

Hybrid Arguments

(1) PRGs which map λ bits to $\lambda+1$ bits can be turned into PRGs which map λ bits to $\lambda+p(\lambda)$ bits for some arbitrary polynomial p.

(2) Length-doubling PRGs can be turned into pseudorandom functions via the Goldreich-Goldwasser-Micali construction.

While (1) and (2) are interesting statements of their own right, they also provide two examples of the proof technique known as *hybrid arguments*.

Search-to-decision

(3) We know that bijective, length-preserving OWFs can be turned into PRGs via the Goldreich-Levin hardcore bit. We prove that the Goldreich-Levin construction is, indeed, a hardcore bit.

(4) We prove that, in general, OWFs can be turned into PRGs. This was originally proved by Hastad, Impagliazzo, Levin and Luby, but we present a simpler proof (and more efficient construction) by Vadhan and Zheng `https://eccc.weizmann.ac.il/report/2011/141/`.

(3) and (4) are interest and useful also because they show us how to turn a *distinguishing* algorithm, which only gives us 1 bit (in cryptography we refer to decision algorithms as distinguishing algorithms[2]) into a *search* algorithm (which gives us a string, namely a pre-image).

[1] For a δ-distributional one-way functions, any PPT adversary cannot find *uniformly random* pre-images, there is always a δ-gap in the distribution returned by the inverter and the actual uniform distribution over the pre-images.

[2] Technically, a distinguishing algorithm isn't a decision algorithm, since we require that a distinguishing algorithm be correct in the majority of instances, not necessarily all of them. For example, a distinguisher for a PRG is considered successful, if it outputs 1 with probability more than 1/2 (it need not be 1).

Separation results

(5) We already saw that one-way functions do not imply collision-resistant hash-functions. Which other things are *not* implied by one-way functions, i.e., outside of MiniCrypt. Can we build, e.g., public-key encryption from one-way functions? We do not know a definite answer, but the oracle separation by Impagliazzo and Rudich gives us some indication. We prove this oracle separation result.

(6) While (5) was about stronger cryptography, but what about weaker cryptography? Can we build one-way functions based on weaker assumptions? We will see a separation results that shows that it might be hard to build one-way functions merely from the assumption that NP is not equal to P.

Separation results such as (5) and (6) are a way for us to check whether "standard" or "black-box" techniques can solve our problems. Note that the notion of standard/black-box techniques can be formalized, but they often remain a little fuzzy/dependent on context. Another nice topic of hardness amplification is trying to build a one-way function from weaker notions of one-wayness. We saw this result recently, before Lecture 7.

2 Overview over Lecture 7

In Lecture 9 and Lecture 10, we will see how we can obtain a PRG from a one-way function, and there, we will focus on getting a PRG which gives us *one* bit of stretch, i.e., maps λ to $\lambda + 1$ bits. Why is this useful? Why all the effort just for a single additional bit of (pseudo-)randomness?
Given a pseudorandom generator (PRG) which maps λ bits to $\lambda + 1$ bits, how can be build a PRG which maps n bits to $n + p(\lambda)$ many bits for some arbitrary polynomial $p^(\lambda)$? The construction, essentially, consists of iterating the pseudorandom generator many times (see Figure 1 and Figure 2). This comes at the complication that we need to use the security of the pseudorandom generator many times in the security argument. This technique is known as *game-hopping* or *hybrid argument*, a main technique that we use this week and next week.

Further References If you would like to read further references on the hybrid technique, see `https://www.youtube.com/watch?v=TQY5AsZXuqw` for a short (6 min.) video explanation of (one version of) the hybrid argument by myself or Appendix B of `https://eprint.iacr.org/2018/306` or the proof of Theorem 3.2.6 in *Foundations of Cryptography I*.

Outline for today's lecture

(1) We recall the definition of PRGs.

(2) We recall how, from a PRG g with stretch $s(\lambda) = 1$, we can construct a PRG G with stretch $s(\lambda) = p(\lambda)$ for some arbitrary polynomial p in λ.

(3) We give a high-level overview over the proof and introduce the notion of hybrid arguments/game-hops as a general proof structure.

(4) We look into the proof.

3 Definition and Construction

Definition 3.1. Let s be a function $s : \mathbb{N} \to \mathbb{N}$ such that for all $\lambda \in \mathbb{N}$, $s(\lambda) \geq 1$. Let $g : \{0,1\}^* \to \{0,1\}^*$ be efficiently computable such that for all $x \in \{0,1\}^*$, $|g(x)| = |x|+s(|x|)$. g is a secure PRG (or *PR-secure*) if the real and ideal games Gprg_g^0 and Gprg_s^1 are computationally indistinguishable, i.e., the advantage

$$\mathbf{Adv}_{\mathcal{A}}^{\mathtt{Gprg}_g^0,\mathtt{Gprg}_s^1}(\lambda) := \left|\Pr\left[1 = \mathcal{A} \to \mathtt{Gprg}_g^0\right] - \Pr\left[1 = \mathcal{A} \to \mathtt{Gprg}_s^1\right]\right|$$

is negligible in λ.

Gprg_g^0	Gprg_s^1
Parameters	Parameters
λ: security par.	λ: security par.
$s(\lambda)$: length-exp.	$s(\lambda)$: length-exp.
g: function	
State	State
y: image value	y: random value
$\mathsf{SAMPLE}()$	$\mathsf{SAMPLE}()$
assert $y = \perp$	**assert** $y = \perp$
$x \leftarrow_\$ \{0,1\}^\lambda$	
$y \leftarrow g(x)$	$y \leftarrow_\$ \{0,1\}^{\lambda+s(\lambda)}$
return y	**return** y

Theorem (PRG Length-expansion)**.** Let g be a pseudorandom generator (PRG) with stretch $s(\lambda) = 1$, i.e., $|g(x)| = |x| + 1$. Let $p(\lambda)$ be any polynomial. Then, G is a PRG with stretch $s(\lambda) = p(\lambda)$, i.e, $|G(x)| = |x| + s(|x|)$.
In particular, there is a PPT reduction $\mathcal{R}$ such that for all PPT adversaries $\mathcal{A}$, it holds that

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

i.e.,

$$\left|\Pr\left[1 = \mathcal{A} \overset{\mathsf{SAMPLE}}{\to} \mathtt{Gprg}_G^0\right] - \Pr\left[1 = \mathcal{A} \overset{\mathsf{SAMPLE}}{\to} \mathtt{Gprg}_{p(\lambda)}^1\right]\right|$$
$$= s(\lambda) \cdot$$
$$\left|\Pr\left[1 = \mathcal{A} \overset{\mathsf{SAMPLE}}{\to} \mathcal{R} \overset{\mathsf{SAMPLE}}{\to} \mathtt{Gprg}_g^0\right] - \Pr\left[1 = \mathcal{A} \overset{\mathsf{SAMPLE}}{\to} \mathcal{R} \overset{\mathsf{SAMPLE}}{\to} \mathtt{Gprg}_g^1\right]\right|$$

Before we turn to the proof overview, let us quickly recall notation. We will consider the PRG games for g with $s(\lambda) = 1$ as well as for G with $s(\lambda) = p(\lambda)$.

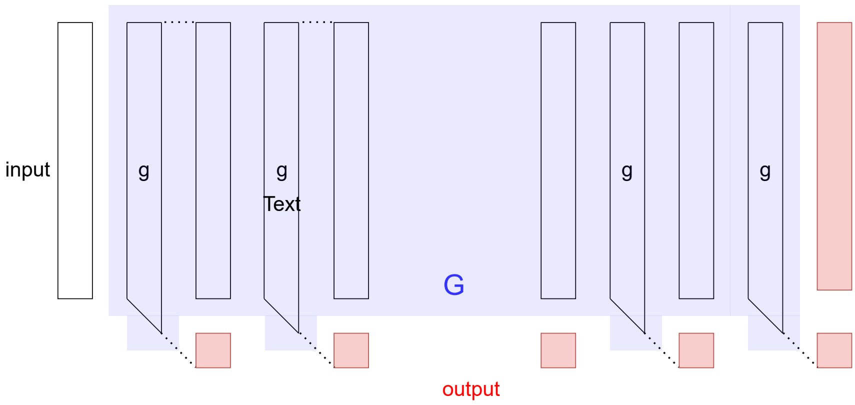

Figure 1: PRG Length-Expansion Construction

$G(x)$

$s_0 \leftarrow x$
for i **from** 1 **to** $s(|x|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
($y[i]$ denotes 1 bit.)
$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$
return y

Gprg^0_G

SAMPLE()

assert $y = \perp$
$x \leftarrow\$ \{0,1\}^\lambda$
$s_0 \leftarrow x$
for i **from** 1 **to** $s(|x|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
($y[i]$ denotes 1 bit.)
$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$
return y

$\mathtt{H}_j$

SAMPLE

assert $y \neq \perp$
for i **from** 1 **to** j :
  $y[i] \leftarrow\$ \{0,1\}$
$s_j \leftarrow\$ \{0,1\}^\lambda$
for i **from** $j{+}1$ **to** $p(|x|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
($y[i]$ denotes 1 bit.)
$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$
return y

$\mathcal{R}_j$

SAMPLE

assert $y \neq \perp$
for i **from** 1 **to** $j-1$:
  $y[i] \leftarrow\$ \{0,1\}$
$s_j||y[j] \leftarrow$ SAMPLE
for i **from** $j{+}1$ **to** $p(|x|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
($y[i]$ denotes 1 bit.)
$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$
return y

Figure 2: Construction G (left), real game $\mathtt{Gprg}^0_G$ (middle left), hybrid game $\mathtt{H}_j$ (middle right), reduction $\mathcal{R}_j$ (right).

We now just re-write these games with these variables. We also inline the code of G into $\texttt{Gprg}_G^0$ in the right-most game below:

$\texttt{Gprg}_g^0$	$\texttt{Gprg}_{s(\lambda)=1}^1$	$\texttt{Gprg}_G^0$	$\texttt{Gprg}_{s(\lambda)=p(\lambda)}^1$	$\texttt{Gprg}_G^0$
SAMPLE()	SAMPLE()	SAMPLE()	SAMPLE()	SAMPLE()
assert $y = \perp$	**assert** $y = \perp$	**assert** $y = \perp$	**assert** $y = \perp$	**assert** $y = \perp$
$x \leftarrow_\$ \{0,1\}^\lambda$		$x \leftarrow_\$ \{0,1\}^\lambda$		$x \leftarrow_\$ \{0,1\}^\lambda$
$y \leftarrow g(x)$	$y \leftarrow_\$ \{0,1\}^{\lambda+1}$	$y \leftarrow G(x)$	$y \leftarrow_\$ \{0,1\}^{\lambda+p(\lambda)}$	$s_0 \leftarrow x$
return y	**return** y	**return** y	**return** y	**for** i **from** 1 **to** $s(\|x\|)$:
				$\quad s_i \| \| y[i] \leftarrow g(s_{i-1})$
				($y[i]$ denotes 1 bit.)
				$y \leftarrow y[1] \| \| .. \| \| y[p(\lambda)] \| \| s_{p(\lambda)}$
				return y

In this proof, we will rely on a *hybrid* argument. I.e., we start with game $\texttt{H}_0$...and prove the following claims.
Now, define $\mathcal{R}$ as the reduction which samples a uniformly random $j \leftarrow_\$ \{1,..,p(\lambda)\}$.
Then, we can derive Theorem **??** from Claim **??** and Claim **??** as follows:

$$
\begin{aligned}
&\left|\Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \texttt{Gprg}_G^0\right] - \Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \texttt{Gprg}_{p(\lambda)}^1\right]\right| \\
=& \left|\Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \texttt{H}_0\right] - \Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \texttt{H}_{p(\lambda)}\right]\right| \quad \text{(by Claim 2)} \\
=& \left|\sum_{j=1}^{p(\lambda)} \Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \texttt{H}_{j-1}\right] - \Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \texttt{H}_j\right]\right| \quad \text{(telescopic sum)} \\
=& p(\lambda) \cdot \left|\left(\sum_{j=1}^{p(\lambda)} \frac{1}{p(\lambda)} \cdot \Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \texttt{H}_{j-1}\right]\right) - \left(\sum_{j=1}^{p(\lambda)} \frac{1}{p(\lambda)} \cdot \Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \texttt{H}_j\right]\right)\right| \quad \text{(multiply by 1)} \\
=& p(\lambda) \cdot \left|\left(\sum_{j=1}^{p(\lambda)} \frac{1}{p(\lambda)} \cdot \Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \mathcal{R}_j \overset{\text{SAMPLE}}{\rightarrow} \texttt{Gprg}_g^0\right]\right) - \left(\sum_{j=1}^{p(\lambda)} \frac{1}{p(\lambda)} \cdot \Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \mathcal{R}_j \overset{\text{SAMPLE}}{\rightarrow} \texttt{Gprg}_g^1\right]\right)\right| \\
=& p(\lambda) \cdot \left|\Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \mathcal{R} \overset{\text{SAMPLE}}{\rightarrow} \texttt{Gprg}_g^0\right] - \Pr\left[1 = \mathcal{A} \overset{\text{SAMPLE}}{\rightarrow} \mathcal{R} \overset{\text{SAMPLE}}{\rightarrow} \texttt{Gprg}_g^1\right]\right|
\end{aligned}
$$

$+H_0 - H_1 + H_1 - H_2 + H_2 - H_3 + H_3 \ldots - H_{p(\lambda)}$

$H_{j-1} - H_j$

$\mathcal{R}$: $j \leftarrow \$ \{1, .., p(\lambda)\}$
run $\mathcal{R}_j$

$1/p(\lambda)$

Claim 1 (Extreme Hybrids).

$$\mathtt{H}_0 \overset{\text{code}}{\equiv} \mathtt{Gprg}^0_G$$
$$\mathtt{H}_{p(\lambda)} \overset{\text{code}}{\equiv} \mathtt{Gprg}^1_{s(\lambda)=p(\lambda)}$$

$\underline{\underline{\mathtt{H}_0}}$

SAMPLE

assert $y \neq \perp$

~~**for** i **from** 1 **to** 0 :~~
~~$y[i] \leftarrow\$ \{0,1\}$~~

$s_0 \leftarrow\$ \{0,1\}^\lambda$

for i **from** $\boxed{0}+1$ **to** $p(|x|)$:

$s_i||y[i] \leftarrow g(s_{i-1})$

($y[i]$ denotes 1 bit.)

$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$

return y

$\underline{\underline{\mathtt{H}_0}}$

SAMPLE

assert $y \neq \perp$

$s_0 \leftarrow\$ \{0,1\}^\lambda$

for i **from** 1 **to** $p(|x|)$:

$s_i||y[i] \leftarrow g(s_{i-1})$

($y[i]$ denotes 1 bit.)

$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$

return y

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

SAMPLE()

assert $y = \perp$

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

$s_0 \leftarrow x$

for i **from** 1 **to** $p(|x|)$:

$s_i||y[i] \leftarrow g(s_{i-1})$

($y[i]$ denotes 1 bit.)

$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$

return y

$\underline{\underline{\mathtt{H}_{p(\lambda)}}}$

SAMPLE

assert $y \neq \perp$

for i **from** 1 **to** $p(\lambda)$:

$y[i] \leftarrow\$ \{0,1\}$

$s_{p(\lambda)} \leftarrow\$ \{0,1\}^\lambda$

~~**for** i **from** $p(\lambda)+1$ **to** $p(|x|)$:~~
~~$s_i||y[i] \leftarrow g(s_{i-1})$~~
~~*($y[i]$ denotes 1 bit.)*~~

$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$

return y

$\underline{\underline{\mathtt{H}_{p(\lambda)}}}$

SAMPLE

assert $y \neq \perp$

for i **from** 1 **to** $p(\lambda)$:

$y[i] \leftarrow\$ \{0,1\}$

$s_{p(\lambda)} \leftarrow\$ \{0,1\}^\lambda$

$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$

return y

$p(\lambda)$ λ

$\underline{\underline{\mathtt{Gprg}^1_{p(\lambda)}}}$

SAMPLE

assert $y \neq \perp$

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

return y

Claim 2 (Reduction between hybrids)**.**

$$\text{For all } j \in \{1,..,p(\lambda)\} : \mathtt{H}_{j-1} \overset{\text{code}}{\equiv} \mathcal{R}_j \overset{\mathsf{SAMPLE}}{\rightarrow} \mathtt{Gprg}_g^0$$

$$\text{For all } j \in \{1,..,p(\lambda)\} : \mathtt{H}_j \overset{\text{code}}{\equiv} \mathcal{R}_j \overset{\mathsf{SAMPLE}}{\rightarrow} \mathtt{Gprg}^1_{s(\lambda)=1}$$

$\underline{\underline{\mathtt{H}_{j-1}}}$

SAMPLE

assert $y \neq \perp$

for i **from** 1 **to** $j-1$:
  $y[i] \leftarrow\$ \{0,1\}$
$s_{j-1} \leftarrow\$ \{0,1\}^\lambda$

for i **from** j **to** $p(|\lambda|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$
return y

$\underline{\underline{\mathtt{H}_{j-1}}}$

SAMPLE

assert $y \neq \perp$

for i **from** 1 **to** $j-1$:
  $y[i] \leftarrow\$ \{0,1\}$
$s_{j-1} \leftarrow\$ \{0,1\}^\lambda$
$s_j||y[j] \leftarrow g(s_{j-1})$
for i **from** $j{+}1$ **to** $p(|\lambda|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$
return y

$\underline{\underline{\mathcal{R}_j \overset{\mathsf{SAMPLE}}{\rightarrow} \mathtt{Gprg}_g^0}}$

SAMPLE

assert $y \neq \perp$

for i **from** 1 **to** $j-1$:
  $y[i] \leftarrow\$ \{0,1\}$
$x \leftarrow\$ \{0,1\}^\lambda$
$s_j||y[j] \leftarrow g(x)$
for i **from** $j{+}1$ **to** $p(|\lambda|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$
return y

$\underline{\underline{\mathcal{R}_j}}$

SAMPLE

assert $y \neq \perp$

for i **from** 1 **to** $j-1$:
  $y[i] \leftarrow\$ \{0,1\}$
$s_j||y[j] \leftarrow \mathsf{SAMPLE}$

for i **from** $j{+}1$ **to** $p(|\lambda|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$
return y

$\underline{\underline{\mathtt{H}_j}}$

SAMPLE

assert $y \neq \perp$

for i **from** 1 **to** j :
  $y[i] \leftarrow\$ \{0,1\}$
$s_j \leftarrow\$ \{0,1\}^\lambda$
for i **from** $j{+}1$ **to** $p(|\lambda|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$
return y

$\underline{\underline{\mathcal{R}_j \overset{\mathsf{SAMPLE}}{\rightarrow} \mathtt{Gprg}^1_{s(\lambda)=1}}}$

SAMPLE

assert $y \neq \perp$

for i **from** 1 **to** $j-1$:
  $y[i] \leftarrow\$ \{0,1\}$
$s_j||y[j] \leftarrow\$ \{0,1\}^{\lambda+1}$
for i **from** $j{+}1$ **to** $s(|\lambda|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
$y \leftarrow y[1]||..||y[s(\lambda)]$
return y

$\underline{\underline{\mathcal{R}_j}}$

SAMPLE

assert $y \neq \perp$

for i **from** 1 **to** $j-1$:
  $y[i] \leftarrow\$ \{0,1\}$
$s_j||y[j] \leftarrow \mathsf{SAMPLE}$
for i **from** $j{+}1$ **to** $p(|\lambda|)$:
  $s_i||y[i] \leftarrow g(s_{i-1})$
$y \leftarrow y[1]||..||y[p(\lambda)]||s_{p(\lambda)}$
return y