

Solutions to homework 4

1 “One-bit-streching” PRG \Rightarrow “polynomially-streching” PRG

Assume that G is a PRG which stretches input by only one bit, i.e. for all inputs x , the length $|G(x)|$, of the output of G on x is equal to $|x| + 1$.

1.1

For *any* polynomial $p(\cdot)$, use the 1-bit stretching PRG G to construct a PRG G' which stretches the (random) k -bit input into a (pseudorandom) output of length $p(k)$. Prove that your construction G' is indeed a PRG if G is a PRG.

Hint(s): First try to construct a two-bit stretching G' , i.e. do it for $p(k) = k + 2$. (Note that in the subsection below you have some *wrong* ways of making the 2-bit stretching PRG. I think that all ways where you try to use G just once will fail, and to get $(2+k)$ -bit output you need to use G twice.) If you do get it for 2-bit stretching PRG, chances are that your construction generalizes to any polynomial number of extra bits, and that you can prove this generalized construction using the proof you did for the 2-bit case and induction.

And how can you prove that your construction for G' is secure? You can try to prove this by contradiction, i.e. assume that G' is not a PRG, i.e. that there exists a PPT adversary which distinguishes outputs of G' from random strings, and try to use that adversary to attack the PRG G itself, which is supposed to be secure.

You might also try a direct proof (this could in fact be easier!) to argue why the distribution $\{G'(x)\}_{x \leftarrow \{0,1\}^k}$ is computationally indistinguishable from distribution $\{r\}_{r \leftarrow \{0,1\}^{k+2}}$. Recall that the fact that G is a good (1-bit stretching) PRG can be phrased as

$$\{G(x)\}_{x \leftarrow \{0,1\}^k} \approx \{r\}_{r \leftarrow \{0,1\}^{k+1}}$$

(where “ \approx ” stands for “computationally indistinguishable”).

In coming up with the direct proof, you can use the following two lemmas, which we used recently in lectures:

Lemma 1 *If X, Z are two computationally indistinguishable distributions, i.e. $\{s\}_{s \leftarrow X} \approx \{s\}_{s \leftarrow Z}$, and $f(\cdot)$ is a PPT algorithm, then $\{f(s)\}_{s \leftarrow X} \approx \{f(s)\}_{s \leftarrow Z}$.*

Using a simplified notation: If $\{X\} \approx \{Y\}$ and f is PPT then $\{f(X)\} \approx \{f(Y)\}$.

Lemma 2 (Hybrid Lemma) *If X_1, \dots, X_n are distributions s.t. $\{X_i\} \approx \{X_{i+1}\}$ for every $i = 1, \dots, n - 1$, and n is polynomial in the security parameter, then $\{X_1\} \approx \{X_n\}$.*

Solution: Let’s first do a 2-bit stretching G' . Namely, let’s have

$$G'(x) = G(G(x))$$

Clearly, $|G'(x)| = |x| + 2$ for all x . Now we’ll show that G' is a PRG. By assumption on G , we have:

$$\{G(x)\}_{x \leftarrow \{0,1\}^k} \approx U_{k+1} \tag{1}$$

(where U_k is a uniform distribution of binary strings of length k , i.e. $U_k = \{r\}_{r \leftarrow \{0,1\}^k}$).

Moreover, since G is a good 1-bit stretching PRG when executed on strings of any length, it works just as well for inputs of length $k + 1$, and therefore:

$$\{G(r)\}_{r \leftarrow \{0,1\}^{k+1}} \approx U_{k+2} \quad (2)$$

By our lemma, taking $X = \{G(x)\}_{x \leftarrow \{0,1\}^k}$, $Z = U_{k+1}$, and $f(s) = G(s)$, equation (1) leads to the following equation:

$$\{G(G(x))\}_{x \leftarrow \{0,1\}^k} \approx \{G(r)\}_{r \leftarrow \{0,1\}^{k+1}} \quad (3)$$

If we denote $X_1 = \{G(G(x))\}_{x \leftarrow \{0,1\}^k}$, $X_2 = \{G(r)\}_{r \leftarrow \{0,1\}^{k+1}}$, and $X_3 = U_{k+2}$, then (3) tells us that $X_1 \approx X_2$, equation (2) tells us that $X_2 \approx X_3$, and using the hybrid lemma tells us that it follows that $X_1 \approx X_3$. In other words we have

$$\{G'(x)\}_{x \leftarrow \{0,1\}^k} = \{G(G(x))\}_{x \leftarrow \{0,1\}^k} \approx U_{k+2}$$

And hence, G' is a PRG.

The extension to any polynomial stretch $p(k)$ is by a repetition of applications of G consecutively, $p(k)$ times, to its output. In other words we can define

$$G^{(1)}(x) = G(x) \quad ; \quad G^{(t+1)}(x) = G(G^{(t)}(x)) \quad \text{for all } t \geq 1$$

And then the $p(k)$ -stretching PRG will be the $G^{(p(k))}$ algorithm. This algorithm is deterministic polynomial time, because $p(k)$ is polynomial in k and the input to each iteration of G is polynomial in k . And to prove that

$$\{G^{(p(k))}(x)\}_{x \leftarrow \{0,1\}^k} \approx U_{p(k)}$$

one can easily extend the above argument by using induction. It is crucial that $p(k)$ is a polynomial in k in this argument. Otherwise the hybrid lemma, used above, will not work. \square

1.2

Here are some *incorrect* constructions of a 2-bit stretching PRG G' from 1-bit stretching PRG G . They all fail in the sense that the resulting algorithm G' could fail to be a secure PRG even if G is a secure PRG. Try to show why that's the case:

Hint: In each case, try to design algorithm G s.t. G is a good PRG but it is designed on purpose so that the G' construction which uses G , fails to be a PRG. In other words, design a PRG G which makes outputs of G' easily distinguishable from random strings. How can you make such a G ? The easiest way is to take yet another PRG, say \tilde{G} , and construct G from \tilde{G} in such a way that: (1) G is a PRG if \tilde{G} is, and (2) G is designed so that to make the G' construction easily breakable (i.e. makes G' distinguishable from random). To construct one PRG from another you might resort to the same two lemmas above.

- $G'(x) = G(0x)$

Solution: Take any good 1-bit stretching PRG \bar{G} , and define $G(x) = [x_1 \mid \bar{G}(x_{[2..k]})]$, where $k = |x|$, x_1 is the first bit of x , and $x_{[2..k]}$ are the remaining bits of x , i.e. $x = [x_1 \mid x_{[2..k]}]$. Then G is still a PRG because

$$\begin{aligned} \{G(x)\} &= \{[x_1 \mid \bar{G}(x_{[2..k]})]\}_{x \leftarrow \{0,1\}^k} = \{[b \mid \bar{G}(x')]\}_{b \leftarrow \{0,1\}; x' \leftarrow \{0,1\}^{k-1}} \approx \\ &\approx \{[b \mid r]\}_{b \leftarrow \{0,1\}; r \leftarrow \{0,1\}^k} = U_{k+1} \end{aligned}$$

However, G shows that the $G'(x) = G(0x)$ fails in general, because $G'(0x) = [0 \mid \bar{G}(x_{[2..k]})]$, and hence the outputs of G' always start with 0, and hence they are easily distinguishable from random $k + 2$ -bit strings. \square

- $G'(x) = [G(x) \mid b_{\oplus}(x)]$ where $b_{\oplus}(x)$ denotes an exclusive or of all bits of x

Solution: We make a similar argument here. The idea is that if $G'(x)$ “leaks” the $b_{\oplus}(x)$ bit function of x , then if we make $G(x)$ itself include the same bit function in its output, the output of G' will have this bit twice, and hence will be distinguishable from random strings. How to make such G ? Take a good 1-bit stretching PRG \bar{G} and define $G(x) = [\bar{G}(x_{[2..k]}) \mid b_{\oplus}(x)]$, where $|x| = k$. Note that $|G(x)| = |\bar{G}(x_{[2..k]})| + 1 = ((k - 1) + 1) + 1 = k + 1$, so G is 1-bit stretching if \bar{G} is. Also outputs of G are indistinguishable from random because note that $b_{\oplus}(x) = x_1 \oplus b_{\oplus}(x_{[2..k]})$, and hence for randomly chosen bit x_1 , $b_{\oplus}(x)$ is a random bit distributed independently from $x_{[2..k]}$. In other words,

$$\begin{aligned} \{G(x)\} &= \{[\bar{G}(x_{[2..k]}) \mid b_{\oplus}(x)]\}_{x \leftarrow \{0,1\}^k} = \{[\bar{G}(x') \mid b]\}_{x' \leftarrow \{0,1\}^{k-1}; b \leftarrow \{0,1\}} \approx \\ &\approx \{[r \mid b]\}_{r \leftarrow \{0,1\}^k; b \leftarrow \{0,1\}} = U_{k+1} \end{aligned}$$

However, as in the previous example, G shows that $G'(x) = [G(x) \mid b_{\oplus}(x)]$ construction fails in general, because $G'(x) = [\bar{G}(x_{[2..k]}) \mid b_{\oplus}(x) \mid b_{\oplus}(x)]$, and hence the last two bits of outputs of G' are always the same. \square

2 PRGs imply OWFs

We showed in class that existence of a One-Way *Permutation* implies a PRG.

2.1

Show that existence of a PRG implies the existence of a One-Way *Function*. In fact, simply show that a PRG G which is stretching its inputs by some polynomial $p(\tau)$ amount, for large-enough $p(\tau)$, must be a one-way function.

Hint: If you are lost, compare this to the previous homework, where we showed that any encryption implies one-way functions.

Solution: Assume that there exists a PRG G . In the worst case, G stretches the input by only 1 bit, but we have seen in exercise 1.1 above, that for any polynomial $p(\cdot)$, existence of such G implies existence of PRG G' , s.t. $|G'(x)| = p(|x|)$, for all x . So take $p(\tau) = 2\tau$. I claim that G' which is a PRG s.t. $|G'(x)| = 2|x|$ must be a one-way function itself. (Recall that a PRG is a *deterministic* algorithm, and hence a function.)

How to show that? Assume otherwise, i.e. assume that there exists a PPT adversary A which breaks the one-wayness property of G' . We can use this A to create an efficient attack A' against the pseudorandomness of $G' : \{0, 1\}^\tau \leftarrow \{0, 1\}^{2\tau}$. Let algorithm A' , on input $y \in \{0, 1\}^{2\tau}$, get $z = A(y)$ by running A on y , and test if $G'(z) = y$. If $G'(z) = y$, A' decides that y is pseudorandom and outputs 1. Otherwise A' decides that y is random and outputs 0. By assumption on A , the probability

$$\text{Prob}[G'(z) = y \mid x \leftarrow \{0, 1\}^\tau; y = G'(x); z = A(y)] \geq \epsilon(\tau)$$

where $\epsilon(\tau)$ is some non-negligible function of τ . By the above description of A' in terms of A we have:

$$\begin{aligned} & \text{Prob}[A'(y) = 1 \mid x \leftarrow \{0, 1\}^\tau; y = G'(x)] = \\ & = \text{Prob}[G'(z) = y \mid x \leftarrow \{0, 1\}^\tau; y = G'(x); z = A(y)] \end{aligned}$$

And therefore,

$$\text{Prob}[A'(y) = 1 \mid x \leftarrow \{0, 1\}^\tau; y = G'(x)] \geq \epsilon(\tau) \quad (4)$$

So, on pseudorandom y 's, A' answers "1" with probability at least $\epsilon(\tau)$.

$$\text{Prob}[A'(y) = 1 \mid y \leftarrow \{0, 1\}^{2\tau}] \leq 1/2^\tau \quad (5)$$

because A' answers 1 only if $A(y)$ returns z s.t. $G'(z) = y$. But note that the range of $G' : \{0, 1\}^\tau \rightarrow \{0, 1\}^{2\tau}$ can only have 2^τ elements, and therefore if you pick random $y \leftarrow \{0, 1\}^{2\tau}$, the probability that there *exists* $z \in \{0, 1\}^\tau$ s.t. $y = G'(z)$ is at most $2^\tau / 2^{2\tau} = 1/2^\tau$.

Taking together equations (4) and (5) we get that G' must be an insecure PRG, because we have an efficient A' s.t.

$$\text{Prob}[A'(y) = 1 \mid x \leftarrow \{0, 1\}^\tau; y = G'(x)] - \text{Prob}[A'(y) = 1 \mid y \leftarrow \{0, 1\}^{2\tau}] \geq \epsilon(\tau) - 1/2^\tau$$

which is non negligible. Since this contradicts our assumption it follows that G' must indeed be a OWF. \square

2.2

Show that the converse is not true, i.e. show that there are one-way functions which are not PRG's. (You should assume here that *some* one-way functions, like modular exponentiation or RSA, do exist.)

Clarification: One very simple (and good!) answer is that a one-way function does not have to stretch the inputs, i.e. if f is OWF we might have $|f(x)| \leq |x|$ for all x , and hence immediately it is not a good PRG. In fact, for all one-way *permutations* (which are also one way functions), $|f(x)| = |x|$ for all x , and this is the case for the two one-way functions we most often talk about, the modular exponentiation and the RSA permutation. However, show that f can be both one-way *and* stretching, and still not a PRG.

Solution: There are many good answers. For example, if f is a OWF then so is $f'(x) = [f(x)|0^k]$, i.e. $f(x)$ concatenated with k zeroes, for any k polynomial in the security parameter. f' remains a OWF if f is, and for large-enough k we can always make f' to be "stretching", i.e. $|f'(x)| > |x|$ for all x . And clearly, f' is not a PRG because its outputs, always ending in a string of zeroes, are easily distinguishable from random strings.

Another example is to make $f'(x) = [f(x) \mid f(x) \mid \dots]$, enough times so that $|f'(x)| > |x|$ for all x . Again, f' is a OWF if f is but clearly distinguishable from random. \square