

CAS CS 538. Problem Set 2

Note: The answers below must be *proven*, using one of the definitions of pseudorandomness used in class (see notes3-5). Also, once again, be concise (and as formal/precise as possible).

Problem 1. Suppose an algorithm G is a pseudorandom generator. Let \bar{G} be the following algorithm: on input seed s , run $G(s)$ to get w , then negate every bit of w to get \bar{w} (i.e., for bit i , $\bar{w}_i = 1 - w_i$), and output the result. Prove that \bar{G} is also a pseudorandom generator.

Problem 2. Suppose algorithms G_1 and G_2 are pseudorandom generators. Let G_3 be the following algorithm: on input s , G_3 runs $G_1(s)$ to get w_1 , runs $G_2(s)$ to get w_2 , and outputs the concatenation of the two strings: $w_3 = w_1 \circ w_2$. Show that G_3 is *not* necessarily a pseudo-random generator. (Hint: it may be helpful to use what you proved in Problem 1.)

Problem 3. In the previous problem, we saw an *insecure* way to combine two pseudorandom generators: run them on the same seed. Here we will show that running them on two *independent* seeds is *secure*.

Suppose algorithms G_1 and G_2 are pseudorandom generators. Let G_3 be the following algorithm: on input s_3 (assume length of s_3 is even), G_3 splits s_3 in half to get two strings s_1 and s_2 of half the length. Then G_3 runs $G_1(s_1)$ to get w_1 , runs $G_2(s_2)$ to get w_2 , and outputs the concatenation of the two strings: $w_3 = w_1 \circ w_2$. Show G_3 is a pseudorandom generator. (Hint: suppose it's not. Then there is a distinguisher that can tell w_3 from random. Use a "hybrid" argument—unlike the complicated one we did in class, where we had many intermediate points, here you only need one intermediate point.)

Problem 4. Do the previous two problems using bitwise XOR instead of concatenation.

Problem 5. Let G_{+1} be a pseudorandom generator, which given a random seed s of length k outputs a pseudorandom string $G_{+1}(s)$ of length $k + 1$. Using G_{+1} , construct a length doubling pseudorandom generator $G_{\times 2}$ (on input seed s of length k outputting pseudorandom string $G_{\times 2}(s)$ of length $2k$). The complexity of $G_{\times 2}$ should be k invocations of G_{+1} , (*all*) on input of length k .

Then construct a generator F , which when invoked as $F(s, -1)$ outputs a pseudorandom string of length k^3 (where k is the length of random seed s), but when invoked as $F(s, i)$ for $0 \leq i < k^3$, outputs the i -th bit of the above string. The complexity of $F(s, i)$ computation must be $O(\log k)$ computations of $G_{\times 2}$ on length k input.

(Do not forget to prove that both of your generators $G_{\times 2}$ and F are indeed pseudorandom.)