

CAS CS 538. Problem Set 3

Due in class Wednesday, September 29, 2010, *before* the start of lecture

Problem 1. (20 points) Prove that if the discrete logarithm assumption (formally described in Section 2.3 of the notes) holds, then it is hard to compute x given g^{xy} and g^y . Formally, prove that if the discrete logarithm assumption holds, then for any poly-time algorithm A , there exists a negligible function η such that, if you generate random k -bit p and its generator g and select a random $x, y \in \mathbb{Z}_p^*$, $\Pr[A(p, g, g^{xy} \bmod p, g^y \bmod p) = x] \leq \eta(k)$. Do so via a reduction.

Note: The answers below must be *proven* using one of the two definitions of pseudorandomness used in class.

Problem 2.

(a) (20 points)

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 by using a reduction that \bar{G} is also a pseudorandom generator.

(b) (20 points)

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 the previous part.)

Problem 3. (40 points)

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.)