Problem Set 5

Due on Wednesday, April 4, 2012.

Instructions Work the problems below, prepare your answers in electronic form, and submit your solutions using the submit script on the Zoo. Remember to specify "5" for the problem number argument to `submit`.

Problem 1: ElGamal Signature Scheme

Happy Hacker was having trouble understanding the ElGamal signature scheme presented on slide #26, lecture 11. He didn't see why the signer should bother choosing a random y in step 1 and decided instead to simply fix $y = 1$. Help Happy out and explain why this is not a good idea.

Problem 2: ElGamal Authentication

Once Happy understood ElGamal signatures, he was excited to use them for authentication. He wants to send an authenticated message m to Bob so that Bob can verify that m came from him.

Here's his idea. Assume that Happy has an ElGamal signing key (g, p, x) and Bob has the corresponding verification key (g, p, a). We denote the signing algorithm using that key pair by S and the verification algorithm by V.

	Happy		Bob
1.		$\xleftarrow{r}$	Choose random string r.
2.	Compute $s = S_A(r)$	$\xrightarrow{s,m}$	Check $V_A(r, s)$. Accept m as coming from Happy if check succeeds.

(a) Mallory wants to get Bob to accept a message m' of his choosing. Describe in detail how he can do this using a man-in-the-middle attack.

(b) Suggest a way to fix this protocol to thwart Happy's attack. Your suggestion should not use any more rounds of communication nor assume any other encryption system or secret keys. [Hint: Think about using a secure hash function H to somehow "bind" m to the signature.]

Problem 3: Indistinguishability

Happy's roommate, Naive Nelson, is building a pseudorandom sequence generator. He has found a PRSG G that takes a 128-bit seed s and outputs a bit string x of length 1000. Naive wants to generate bit strings of length $\ell = 100,000$, so he creates a new generator G' that works in stages. His idea is to use G repeatedly, obtaining 1000 bits each time. To avoid getting repetitions of the same 1000-bit string, he uses the last 128 bits of each block as a seed for the next block.

Here is his algorithm in greater detail for computing $G'(s)$. $s_0, s_1, \ldots, s_{100}$ are 128-bit strings, where $s_0 = s$. $x_1, \ldots, x_{100}$ comprise a sequence of 1000-bit strings. x_i and s_i are computed from s_{i-1} as follows: $x_i = G(s_{i-1})$. s_i is the last 128 bits of x_i.

Unfortunately, $G'(s)$ is not cryptographically strong. Explain why, and describe a judge J that can distinguish the distribution $G'(S)$ from U. Here, S is the uniform distribution over the seed space, and U is the uniform distribution over binary strings of length ℓ.