CAS CS 538. Problem Set 4
Due via websubmit on Thursday, October 8, 2015, at 11:59pm

Problem 1. (40 points)
On the previous problem set, 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: $w_3 = w_1 \cdot w_2$. 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 with a single intermediate point.)

Problem 2. (30 points) Let $p_1$ and $p_2$ be two primes of length $k$ bits each. Let $n = p_1p_2$, let $c \in Z_n^*$ be a 2$k$-bit value, and let $d$ be a random 2$k$-bit exponent. Assume that the operation of computing $xy \mod z$ takes time exactly $k^2$ when $x, y$ and $z$ are all of length $k$.

(a) How long does it take to compute $m = c^d \mod n$ using the square-and-multiply technique of Problem 1 (assume that half the bits of $d$ are 1)?

(b) Let $c_1 = c \mod p_1$ and $d_1 = d \mod (p_1 - 1)$. Note that $m \equiv c_1^{d_1} \pmod{p_1}$. How long does it take to compute $m_1 = c_1^{d_1} \mod p_1$? Assume that $q_1 = p_1^{-1} \mod p_2$ and $q_2 = p_2^{-1} \mod p_1$ are known. How to compute $m$ faster than in part (a) using the Chinese Remainder Theorem? What is the (approximate) speed-up factor (again, assume half the bits of $d_1$ and half the bits of $d_2$ are 1)?

(c) In fact, you can save a little on the Chinese Remainder Theorem computation, and you don’t even need $q_1$, just $q_2$. Simply compute $m_1 = c^d \mod p_1$ and $m_2 = c^d \mod p_2$ as before, then $h = q_2(m_1 - m_2) \mod p_1$, and then $m$ as $m_2 + hp_2$. Prove that this computation is correct. (Hint: first prove that $m_2 + hp_2$ is congruent to $m_1$ modulo $p_1$ and congruent to $m_2$ modulo $p_2$. Now prove that $0 \leq m_2 + hp_2 < n$. Recall that CRT states that a value satisfying these three conditions is unique—hence it has to be equal to m).

Problem 3. (30 points) Let $p_1$ and $p_2$ be two primes of length $k$ bits each, such that $p_1 \equiv p_2 \equiv 3 \pmod{4}$. Let $u_1 = (p_1 + 1)/4$, and $u_2 = (p_2 + 1)/4$. Recall from Problem 2 that if $s$ is a square modulo $p_1$, then a square root of $s$ is $t = s^{u_1} \mod p_1$.

(a) Prove that $t$ itself is a square modulo $p_1$.

(b) Prove that $2^\ell$-th root of $s$ is equal to $s^{u_1} \mod p_1$ and is itself a square modulo $p_1$.

(c) In light of the above and Problem 2, how (and how much) can you speed up Blum-Goldwasser decryption of an $\ell$-bit message as compared to simply taking square roots $\ell$ times? (Ignore the costs of anything but the modular arithmetic. Assume that $u_1 \mod p_1 - 1$ and $u_2 \mod p_2 - 1$ are computed at key generation and kept with the secret key, so the cost of computing $u_1$ and $u_2$ need not be taken into account.)