Theoretical Foundations of Cryptography **Instructor: Chris Peikert**
Georgia Tech, Spring 2011 **Student: YOUR NAME HERE**

Homework 2

This homework is due by the **start of class on February 10** via the course page on T-Square. Start early!

Instructions. Solutions must be typeset in LaTeX (a template for this homework is available on the course web page). Your work will be graded on *correctness*, *clarity*, and *conciseness*. You should only submit work that you believe to be correct; if you cannot solve a problem completely, you will get significantly more partial credit if you clearly identify the gap(s) in your solution. It is good practice to start any long solution with an informal (but accurate) "proof summary" that describes the main idea.

You may collaborate with others on this problem set and consult external sources. However, you must ***write your own solutions*** and ***list your collaborators/sources*** for each problem.

1. *(Number theory.)*
   (a) (6 points) Let N be the product of two distinct n-bit primes, and suppose there is an efficient algorithm $\mathcal{A}$ that computes square roots on a noticeable fraction of quadratic residues mod N:
   $$\Pr_{y \leftarrow \mathbb{QR}_N^*}[\mathcal{A}(N, y) \in \sqrt{y} \bmod N] = \delta \geq 1/\operatorname{poly}(n).$$
   Construct an efficient algorithm $\mathcal{B}$ that, using $\mathcal{A}$ as an oracle, computes the square root of *any* $y \in \mathbb{QR}_N$ with *overwhelming* probability (solely over the random coins of $\mathcal{A}$ and $\mathcal{B}$). That is, for every $y \in \mathbb{QR}_N$, it should be the case that
   $$\Pr[\mathcal{B}^{\mathcal{A}}(N, y) \in \sqrt{y} \bmod N] = 1 - \operatorname{negl}(n).$$
   (b) (4 points) We have seen in class that the "most significant bit" $[x > \frac{p-1}{2}]$ is a hard-core predicate for the modular exponentiation function $f_{p,g}(x) = g^x \bmod p$, where $p > 2$ is prime, g is a generator of $\mathbb{Z}_p^*$, and $x \in \mathbb{Z}_p^* = \{1, \ldots, p-1\}$.
   Prove that the "least significant bit" $[x \text{ is odd}]$ is *not* a hard-core predicate for $f_{p,g}$.

2. *(Indistinguishability potpourri.)* Prove or disprove (giving the simplest counterexample you can find) the following statements. In constructing a counterexample, you may assume the existence of another OWF / PRG / PRF.
   (a) (3 points) For a probability distribution D over Ω and positive integer m, let D^m denote the *product distribution* over Ω^m, obtained by drawing an tuple of m independent samples from D. Let $\mathcal{X} = \{X_n\}$ and $\mathcal{Y} = \{Y_n\}$ be ensembles of distributions that are efficiently sampleable (in PPT), and let $m(n) = \operatorname{poly}(n)$.
   If $\mathcal{X} \overset{c}{\approx} \mathcal{Y}$, then $\{X_n^{m(n)}\} \overset{c}{\approx} \{Y_n^{m(n)}\}$. (Why is it important that X_n, Y_n be efficiently sampleable?)
   (b) (3 points) If an injective (one-to-one) function f has a hard-core predicate h, then f is one-way.
   (c) (4 points) Let G be a PRG with output length $\ell(n) > n$. The function $G'(s) = G(s) \oplus (s|0^{\ell(|s|)-|s|})$ is a PRG, where $|$ denotes concatenation.
   (d) (5 points) A PRG G with output length $\ell(n) = 2n$ is itself a one-way function. (Why do we take $\ell(n) = 2n$ instead of, say, $\ell(n) = n + 1$?)

3. In this problem, you will use a PRG to implement what we'll call a secure "locking" scheme. A locking scheme is a protocol between two players, a locker L and a verifier V. It allows L to lock itself into one of two choices (0 or 1) without V knowing which choice was made, then later reveal its choice. The protocol works in two phases: in the first "locking" phase, L and V exchange some messages, which

result in L being bound to its (secret) choice bit. In the second "unlocking" phase, L reveals its choice bit and some additional information, which allows V to check consistency with the earlier messages.

We define the following model for a locking scheme, in which the locking phase consists of an initial message from the verifier, followed by a response from the locker.

- The verifier $V()$ is a PPT algorithm that takes no input (except for the implicit security parameter 1^n and its random coins) and outputs some message $v \in \{0,1\}^*$.
- The locker $L(\sigma, v; r_L)$ is a PPT algorithm that takes a choice bit $\sigma \in \{0,1\}$, the verifier's initial message v, and random coins r_L, and outputs some message $\ell \in \{0,1\}^*$.

In the unlocking phase, the locker simply reveals σ and r_L, and the verifier checks that $\ell = L(\sigma, v; r_L)$.

(a) (3 points) A secure locking scheme should be "hiding," i.e., a malicious (but computationally *bounded*) verifier V^* should not be able to learn anything about the honest locker L's choice bit σ, no matter what initial message v^* the malicious verifier sent.

Using the notion of indistinguishability, give a formal definition of this hiding property.

(b) (3 points) A secure locking scheme should also be "binding" against even a computationally *unbounded* malicious locker L^*. That is, there should not exist any ℓ^* that can successfully be unlocked as both choice bits $\sigma \in \{0,1\}$, except with negligible probability over the choice of the honest verifier V's initial message v.

Give a formal definition of this binding property.

(c) (3 points) Let G be any length-tripling function, i.e., one for which $|G(x)| = 3|x|$ for every $x \in \{0,1\}^*$. Give an upper bound on the probability, over the choice of a random $3n$-bit string R, that there exist two inputs $x_1, x_2 \in \{0,1\}^n$ such that $G(x_1) \oplus G(x_2) = R$.

(d) (6 points) Let G be a length-tripling PRG (which we have seen can be obtained from any PRG). Use G to construct a secure locking scheme, and prove that it is both hiding and binding according to your definitions.