Homework 6 for 6.875

Teacher: Silvio Micali, TA: Yevgeniy Dodis

Due Date: **Monday, November 30**

Problem 1: One-Way Functions and Hardcore Bits

Recall that the *hardcore bit* of a function f is a predicate of x that is easy to compute given x, but very hard to compute given $f(x)$. For *any* one-way function f, Goldreich-Levin theorem says that a parity of a *random* subset of x's bits is a hardcore bit for f. For *specific* one-way function candidates, we often have a that a particular bit of x is a hardcore bit, e.g. $lsb(x)$ for RSA and $msb(x)$ for exponentiation modulo a prime. This raises a question of whether it is the case that for any one-way function f, some particular bit of x is a hardcore bit.

Show that the general answer is "no". In other words, assuming one-way functions exist, construct a one-way function g such that *no single bit of* x is a hardcore bit for this function. In fact, every bit of x can be predicted with probability at least $\frac{3}{4}$ given $g(x)$. Hence, Goldreich-Levin result is more or less the best we can hope for.

Problem 2: Claw-Free Permutation Pairs

Let $\mathcal{C} = \{(f_i, g_i) : D_i \mapsto D_i \mid i \in I\}$, $F(\mathcal{C}) = \{f_i \mid i \in I\}$, where each f_i, g_i are permutation on some domain D_i. Here I is some infinite index set that is samplable, i.e. there is PPT $G(1^k)$ outputting a random $i \in \{0,1\}^k \cap I$, the number of bits to represent $x \in D_i$ is polynomial in $|i|$, and both g_i and f_i are polynomial-time computable.

Recall that $\mathcal{C}$ is a collection of *claw-free permutation pairs* if no PPT A can find $x, y \in D_i$ s.t. $f_i(x) = g_i(y)$, i.e.

$$\Pr(A(i) = (x,y) \text{ such that } f_i(x) = g_i(y) \mid i \leftarrow G(1^k)) = negl(k)$$

(a) Show that if $\mathcal{C}$ is a collection of claw-free permutation pairs, then $F(\mathcal{C})$ is a collection of one-way permutations.

(b) Under the RSA assumption, construct a family of permutation pairs $\mathcal{C}$ such that $\mathcal{C}$ is *not* claw-free, but $F(\mathcal{C})$ is a collection of one-way permutations.

(c) Recall that a family $\mathcal{H} = \{h_i : \{0,1\}^* \mapsto D_i \mid i \in I\}$ is a family of *collision-free hash functions* if no PPT B can find distinct strings $a, b \in \{0,1\}^*$ mapping to the same value, i.e.

$$\Pr(B(i) = (a,b) \text{ such that } h_i(a) = h_i(b) \mid i \leftarrow G(1^k)) = negl(k)$$

Construct a collision-free family $\mathcal{H}$ of hash functions based on any family $\mathcal{C}$ of claw-free permutation pairs.

Problem 3: Claw-Free Permutations based on Discrete Log

Construct a family of claw-free permutation pairs based of the Discrete Log assumption (remember, your permutations need not be trapdoor). Either directly, or using the previous problem, define a collision-free family of hash functions based on the Discrete Log assumption.

Problem 4: Signature Scheme Secure Under the Random Oracle

Assume the signer, the verifier and the adversary all have access to a *random oracle*, i.e. to a shared random function $\mathcal{O}$, where $\mathcal{O}(x)$ is a truly random string of the same length as x, $\forall x \in \{0,1\}^*$. Assuming trapdoor permutations exist, design a simple signature scheme that is existentially secure against adaptive chosen message attack (under the random oracle). Prove the claimed security of your signature scheme remembering that the adversary has access to the random oracle as well!

This example demonstrates that random oracle is a very powerful primitive to have.