Homework 6

1 Problem 1 (15 points)

The Euler totient function $\phi(N)$ counts the number of elements in $\mathbb{Z}_N^*$, the number of integers in $\{0, 1, \ldots, N-1\}$ that are relatively prime to $N$ (1 is relatively prime to $N$, but 0 is not for $N > 1$).

(a) Show that for a prime power $q = p^a$, that $\phi(q) = (p-1)p^{a-1} = \left(1 - \frac{1}{p}\right)q$

(b) Show that for a positive integer $N$, $\phi(N) = N \times \prod_p \left(1 - \frac{1}{p}\right)$. Here, $p$ varies over the prime factors of $N$, where each $p$ is counted only once. The Chinese Remainder Theorem will be useful here.

2 Problem 2 (20 points)

In class, we saw how to construct a pseudorandom generator from a one-way permutation that had a hardcore bit. In this question, you will see that the permutation requirement is necessary.

Start from a one-way permutation $F$ with a hardcore bit $h$, and construct a one-way function $F'$ with hardcore bit $h'$. $F'$ should have the following properties:

- $F'$ is one-way, assuming the one-wayness of $F$.
- $F'$ has the same domain and co-domain.
- $h'$ is hardcore for $F'$, assuming $h$ is hardcore for $F$.
- If we plug $F'$ into the PRG construction seen in class, the resulting generator will not be a secure PRG

3 Problem 3 (20 points)

In class, we saw that $\text{lsb}$ and $\text{Half}$ were hardcore bits for squaring mod a composite as well as the RSA function. We also saw that $\text{Half}$ was hardcore for discrete exponentiation mod a prime.
(a) Explain why lsb is not a hardcore bit for discrete exponentiation mod a prime. That is, given \( g^x \mod p \) for a prime \( p \) and generator \( g \), explain how to recover the least significant bit of \( x \).

(b) In class, we said that for \( x \in \mathbb{Z}_N \), \( Half \) was analogous to the most significant bit of \( x \). Explain how \( Half \) is different from the most significant bit, and explain why the most significant bit may not be hardcore for any of the one-way functions we saw in class.

4 Problem 4 (30 points)

In class, we saw how to build collision resistance from factoring using quadratic residues. Here, we will show how to build collision resistance from the RSA problem. The function is defined as:

\[
H( (N, e, u), (x, y) ) = x^e u^y \mod N
\]

Here, the key will be a composite integer \( N = pq \) for unknown primes \( p, q \), and prime \( e \) relatively prime to \( \phi(N) = (p - 1)(q - 1) \), and a random integer \( u \in \mathbb{Z}_N^* \). \( x \) in an integer in \( \mathbb{Z}_N^* \), and \( y \in \{0, ..., e - 1\} \).

Suppose you have an adversary \( A \) that can find collisions for \( H \), given \( (N, e, u) \). You will construct an adversary \( B \) that takes as input \( N, e, u \) and computes \( u^{1/e} \mod N \).

(a) First, show how to construct an \( a \in \mathbb{Z}_N \) and \( b \in \mathbb{Z} \) such that \( a^e = u^b \mod N \), and \( 0 < |b| < e \).

(b) Notice that \( a^{1/b} \mod N \) is the \( e \)th root of \( u \). However, we do not know how to efficiently take roots \( \mod N \). Therefore, we need another way to compute the \( e \)th root.

Since \( 0 < |b| < e \) and \( e \) is prime, it must be the case that \( GCD(b, e) = 1 \).

Therefore, there are integers \( s, t \) such that \( bs + te = 1 \), and \( s, t \) can be computed efficiently using the extended Euclidean algorithm.

Use \( a, u, s, t \) to compute the \( e \)th root of \( u \).

(c) Show that the function is no longer collision resistant if \( y \) is allowed to be in the set \( \{0, ..., e\} \).

5 Problem 5 (15 points)

Here, we generalized the fact that computing square roots mod a composite is as hard as factoring. Let \( N = pq \) for unknown primes \( p, q \), and suppose that \( e \) is prime and
divides either \( p - 1 \) or \( q - 1 \), but not both.

Show that computing \( e \)th roots mod \( N \) is as hard as factoring. That is, if you are able to efficiently compute \( e \)th roots, then you can factor \( N \).

[Hint: if \( e \) divides \( p - 1 \), then how many roots does an \( e \)th residue have mod \( p \)? What if \( e \) does not divide \( p - 1 \)?]

**Bonus (10 points):** Extend the above to handle arbitrary \( e \), as long as \( e \) is not relatively prime to \( \phi(N) = (p - 1)(q - 1) \). Note that if \( e \) is relatively prime to \( \phi(N) \), computing \( e \)th roots is the RSA problem, which is not believed to be as hard as factoring.