Homework 5

1 Problem 1 (15 points)

(a) Show that the original version of the decisional Diffie Hellman problem that we saw in class is easy. That is, fix a prime p. You are given

$$(g, g^a \bmod p, g^b \bmod p, h)$$

where g is a random generator of $\mathbb{Z}_p^*$, $a, b \leftarrow \mathbb{Z}_{p-1}$, and h is either $g^{ab} \bmod p$ or $g^c \bmod p$ for a random $c \in \mathbb{Z}_{p-1}$.

Show how to tell whether $h = g^c \bmod p$ or $h = g^{ab} \bmod p$.

(b) Explain why, despite the above attack, the *computational* Diffie Hellman problem might still be hard

(c) Generalize the above attack as follows. Suppose $\mathbb{G}$ is a cyclic finite group of order N, and suppose N has a small factor r. Show that the decisional Diffie Hellman problem can be broken in time proportional to r (and polylogarithmic in N).

(d) A number N is t-smooth if all of its prime factors are at most t. Let $\mathbb{G}$ be a cyclic finite group of order N, where N is the product of distinct prime factors and N is t-smooth for some small t. Show that the discrete log problem is easy in $\mathbb{G}$: given any g and g^a, it is possible efficiently recover a, with a running time that grows with t, but is otherwise logarithmic in N. The Chinese Remainder Theorem will be helpful here.

(e) Show that the discrete log problem is easy over $\mathbb{Z}_N^*$ for any smooth N. That is, if N is t-smooth, you should give an algorithm for the discrete log over $\mathbb{Z}_N^*$ whose running time grows with t, but is otherwise logarithmic in N

Note that the N in part (e) is different from the N in part (d). In part (d), N is the order of the group (the number such that $g^N = 1$), whereas in (e), the order of the group is something very different.

2 Problem 2 (15 points)

Consider the following commitment scheme, built from a group GrGen:

- $\mathsf{Setup}()$: run $(\mathbb{G}, g, p) \leftarrow \mathsf{GrGen}(\lambda)$. We will assume GrGen always produces a prime p. Choose a random $a \in \mathbb{Z}_p$, and compute $h = g^a \in \mathbb{G}$. The commitment key is $k = (g, h)$.
- $\mathsf{Com}((g, h), m; r)$: We will assume the message space is $\mathbb{Z}_p$. Output $g^m h^r$, where r is a random element in $\mathbb{Z}_p$.

(a) Show that the scheme is perfectly hiding.

(b) Show that the scheme is computationally binding, assuming the discrete log problem is hard for $\mathbb{G}$.

3 Problem 3 (20 points)

Let $N = pq$ be the product of two primes. In this problem, we will see that, in addition to p and q being large, it is important that $p - 1$ and $q - 1$ have large prime factors.

(a) Suppose you know an integer r that is a multiple of $p-1$, but not $q-1$. Explain how to factor N. (Hint: what happens when you compute x^r for an integer r?)

(b) Suppose $p - 1$ is t-smooth (recall that this means all of the factors of $p - 1$ are at most t). Explain how to compute an integer r that is a multiple of $p - 1$. Your r should be no larger than about p^t (so its bit length is at most about $t \log_2 p$), and should take time polynomial in t and $\log_2 p$ to compute.

(c) You are not quite done, as your multiple r might also be a multiple of $q - 1$. Explain how to detect this case.

(d) If your r is a multiple of both $p-1$ and $q-1$, then show how to derive a different integer r' that is a multiple of $p - 1$ but not $q - 1$, or vice versa. Assume $p \neq q$ (if $p = q$, we can easily factor by taking square roots).

One option to avoid this attack is to choose p, q to be safe primes, which means that $(p - 1)/2$ and $(q - 1)/2$ are also prime. However, this is not actually necessary, as it turns out that a random large prime p will, with high probability, have $p - 1$ not be smooth.

4 Problem 4 (15 points)

Here, you will show that computing discrete logs mod a composite integer $N = pq$ is as hard as factoring N. In other words, you are given an algorithm A such that given $g, h \in \mathbb{Z}_N^*$, A efficiently computes an integer x such that $g^x \bmod N = h$. (Note that in general $\mathbb{Z}_N^*$ is not cyclic, so the discrete log is not guaranteed to exist. The algorithm for discrete logs is only guaranteed to work when the discrete log exists). You may assume A finds a discrete log with probability 1 when it exists; there is no guarantee that the x outputted by A will lie in any particular range. Show that given A, you can factor N.

To help you, here are some hints:

- Consider running $A(g, h = g^y)$ for a random $g \in \mathbb{Z}_N^*$, and where y is uniform in $[0, 2N]$. Let x be the output of A. Show that $y \neq x$ with noticeable probability, no matter what A does.
- When $x \neq y$, what relationship must x and y satisfy?
- Can you extend the above to compute the order of g, for any $g \in \mathbb{Z}_N*$. Consider running A several times on the same g but different h's.
- Finally, if you could compute the order for any $g \in \mathbb{Z}_N^*$, how does this let you factor N?

5 Problem 5 (10 points)

Let G be a group of prime order q. Show that the discrete log problem can be solved in time $O(\sqrt{q})$. To do so, conisder the hash function $H : \mathbb{Z}_q^2 \mapsto G$ defined as $H(x, y) = g^x \times h^y$, where $h = g^a$ for an unknown discrete log a [1]. Explain how to use the birthday attack on H to compute a in time $O(\sqrt{q})$.

[1] This is like the hash function we saw in class, but abstracted to work with general groups