Homework 6

1 Problem 1 (10 points)

Recall that in the ElGamal cryptosystem, the public key is a pair (g, h) where g is a generator for a group $\mathbb{G}$ of prime order, and $h = g^a$ where $a \in \mathbb{Z}_p$ is the secret key. To encrypt a message $m \in \mathbb{G}$, choose a random r and output $(g^r, h^r \times m)$.

(a) Suppose you have two ElGamal ciphertexts c_0, c_1 encrypting m_0 and m_1, respectively, where m_0, m_1 are unknown. Show how to devise a new ElGamal ciphertext c_2 which encrypts $m_0 \times m_1$. You only know the public key and the ciphertexts; you do not know m_0, m_1, the secret decryption key a, or the encryption randomness.

Thus, ElGamal is multiplicatively homomorphic: given two ciphertexts, it is possible to devise a new ciphertext that encrypts the product of the two plaintexts knowing just the public key.

(b) Let c be an ElGamal ciphertext encrypting an unknown message m. Show how to devise another ElGamal ciphertext c' encrypting m. c' should look like a fresh random ciphertext: its distribution should be the same as if you encrypted m from scratch, and should be independent of c (except that it encrypts the same message). As before, you know only the public key and the ciphertext; you do not know m, a, or the encryption randomness.

Thus, ElGamal is re-randomizeable, meaning you can take a ciphertext, and produce a fresh looking ciphertext that encrypts the same message.

2 Problem 2 (30 points)

Consider the following public key identification scheme. Alice has the secret key, which consists of two large random primes p, q; Bob has the public verification key, which consists of $N = pq$. To authenticate, Bob will send a random quadratic residue $x \in \mathbb{Z}^*$ to Alice. Alice will then respond with a square root y of x. Bob will accept if and only if $y^2 = x \bmod N$.

(a) Explain how Bob chooses the random quadratic residue x, just knowing N.

(b) The above description does not specify how Alice chooses y from among the 4 possible square roots of x. Show that, no matter how Alice chooses y, the protocol is *insecure* against *active* attacks.

(c) Show that the protocol is secure against direct attacks, under the assumption that factoring N is hard. Security should follow, no matter how Alice chooses y.

(d) Suppose Bob is legally required to record all incoming and outgoing messages, in case law enforcement needs a record for an investigation. Alice would like to be able to deny that any identification took place. To do so, she wants to enable Bob to generate, all by himself and *without* interacting with Alice, a transcript that looks exactly like the sequence of messages that would have been exchanged if Alice *had* identified herself to Bob. Bob should be able to do this, despite only knowing N and not p, q. Thus, law enforcement would be unable to tell if Alice actually identified herself, or if Bob simply generated the transcript of the identification by himself.

How should Alice choose the square root y in order to guarantee this property? Explain how Bob samples random-looking transcripts, and prove that Bob's sampled transcripts are distributed identically to actual transcripts of interaction with Alice.

(e) Prove that the identification protocol is secure against eavesdropping attacks, using Alice's method for choosing y from part (d).

3 Problem 3 (30 Points)

A *random self reduction* is a procedure which turns any instance of a problem into a *random* instance of the problem.

For example, let p a prime, and $\mathbb{G}$ a group of order p. Suppose you are given a discrete log instance $(g, h = g^a)$, and suppose that $g \neq 1$ (so that g is a generator). Choose a random $r, s \in \mathbb{Z}_p$ such that $r \neq 0$ and let $g' = g^r$ and $h' = h^r \times g^s$.

(a) Show that, regardless of the original choice of (g, h), (g', h') is a uniformly random pair of group elements conditioned on $g' \neq 1$. Thus (g', h') is a random discrete log instance, independent of (g, h).

(b) Suppose you give someone (g', h') and they give you a discrete log b such that $(g')^b = h'$. Explain how to recover the discrete log of h, namely a, given b, r, s.

A random self reduction therefore shows that, if there is *any* discrete log instance that is hard, a *random* discrete log instance is also hard. This means that there are

no "extra hard" instances, since no instance is harder than the average case. The fact that discrete log admits a random self reduction means we can actually base hardness of the *worst case* version of the problem, rather than an average case problem. It can also be used to amplify the success probability of attacks:

(c) Suppose you have a discrete log adversary A that runs in time t, and solves random discrete log instances with probability ϵ. You know nothing about A except this fact: in particular, maybe A is deterministic, or maybe A is randomized.

Show how to use A to derive an adversary A' which solves discrete log with probability 99/100, but is allowed to run in time about $O(t/\epsilon)$.

Part (c) shows that, for our discrete log assumption, it is actually sufficient to assume that no polynomial time adversary can solve discrete log with high probability. This then implies that no polynomial time adversary can compute discrete logs with inverse polynomial advantage.

(d) Show a random self reduction for DDH. That is, you are given a tuple $(g, u = g^a, v = g^b, w = g^c)$ where g is a generator, a and b are in $\mathbb{Z}_p$, and c is either $ab \bmod p$ or different than ab. We will call the $c = ab$ case a DDH tuple.

You must come up with a new tuple (g', u', v', w') such that:

- If (g, u, v, w) is a DDH tuple, then (g', u', v', w') is a *random* DDH tuple (it should be random even if (g, u, v, w) is a fixed tuple)
- If (g, u, v, w) is *not* a DDH tuple, then (g', u', v', w') is a truly random tuple of group elements, conditioned on g' being a generator and the tuple being *not* a DDH tuple.

The transformation from (g, u, v, w) to (g, u', v', w') must be efficient: you cannot compute discrete logs as part of the transformation.

Note that for part (d), the following simple transformation will not work: (g, u^r, v, w^r). This is a DDH tuple if (g, u, v, w) was a DDH tuple, and isn't a DDH tuple if (g, u, v, w) isn't. However, for a fixed tuple (g, u, v, w), (g, u^r, v, w^r) is not random: for example, the third component is fixed as v. While this transformation won't work, it is a useful starting point to think about.

4 Problem 4 (10 points)

Let $H : \mathcal{K}_\lambda \times \{0,1\}^{m(\lambda)} \to \{0,1\}^{n(\lambda)}$ be a (keyed) hash function. Assume H is collision resistant. Suppose $m(\lambda) \geq n(\lambda) + \lambda$. Show that H is one-way: for any polynomial

time adversary A, there is a negligible ϵ such that

$$\Pr\left[H(k,x') = y : \begin{smallmatrix} k \leftarrow \mathcal{K}_\lambda \\ x \leftarrow \{0,1\}^{m(\lambda)} \\ y \leftarrow H(k,x) \\ x' \leftarrow A(k,y) \end{smallmatrix}\right] < \epsilon(\lambda)$$

5 BONUS Problem 5 (5 points)

Bonus problem: Suppose $m(\lambda) = n(\lambda) + 1$ in Problem 4. Show that H is not necessarily one-way. That is, construct a hash function H such that H is collision resistant, but H is not one-way. You may assume as a building block any collision resistant hash function H'.