I'll soon announce the course policies (exams, late policies, etc.); but the general form of the course will be similar to last quarter.

I want, however, to already emphasize the collaboration policy. It is exactly the same as last quarter. Please follow it in spirit, and to the letter:

You are permitted to collaborate with other students on the assignments but you should read and think about each problem alone for at least a few minutes; its wise to restrict your collaborations to the harder problems, and not to become dependent on this process. In any case, your submission must be written entirely on your own after the collaboration session is over – without reference to any notes made while collaborating. Also, as a matter of intellectual courtesy, indicate who you worked with on a problem. (This need not be reciprocal. If A was stuck on a problem until getting an idea from B who had already solved it, then no annotation is needed on B's assignment.)

1. **(5 points)** Compute $\log_3 693 \bmod 1823$ using Shank's algorithm.

2. **(10 points)** Recall that for prime p and primitive element $g \in Z/p$, the *discrete logarithm problem* is that of computing, for given $y \in (Z/p)*$, an x such that $g^x = y \bmod p$. Can the complexity of the discrete log problem depend on the choice of the primitive element g? Justify your answer.

3. **(10 points)** Professor Evenbetter has devised a public-key encryption system which, he says, combines the advantages of the RSA and Rabin methods: cracking it is as hard as factoring, and ciphertexts (almost always) have unique decodings.

   Evenbetter's method is a slight modification of Rabin's. As in Rabin's system, we begin by picking two large primes p and q, and publicizing their k-bit product n. Now to encrypt a $k/3$-bit message x, append a string r of $k/3$ random bits, and then a string of $k/3$ 0's, to x, forming the string $xr0...0$. The ciphertext is then $(xr0...0)^2 \bmod n$. Decryption is almost always unique because it is very unlikely that of the four square roots of $(xr0...0)^2 \bmod n$, two end with a string of $k/3$ 0's.

   We'll say that an algorithm cracks this system if, for any message x, the algorithm can with high probability output x given the ciphertext $(xr0...0)^2 \bmod n$.

   Critique Evenbetter's cryptosystem. Is it really true that decryptions are almost always unique, and that cracking the cryptosystem is as hard as factoring?

4. **(10 points)** Suppose you're using the El Gamal cryptosystem, and aren't careful in your choice of a random number generator, and wind up using the same number k to transmit two different messages x_1 and x_2.

   Suppose an adversary intercepts both ciphertexts and somehow gets ahold of x_1 (and that x_1 is not 0). Show how he can compute x_2.

5. **(5 points)** The Diffie-Hellman key exchange protocol is vulnerable to a "man in the middle attack": when Moe receives g^x from Alice, he chooses a new x' and sends $g^{x'}$ to Bob; when Moe receives g^y from Bob, he chooses a new y' and sends $g^{y'}$ to Alice. Moe then shares private key $g^{xy'}$ with Alice, and private key $g^{x'y}$ with Bob. Subsequently he can read every message between them while relaying it, without their knowing the difference. (He can also change their messages if he wishes, though of course then he may have to keep doing so to cover up his tracks, and he may get caught if they have some other form of communication.)

   A *Certificate Authority* (CA) is an organization whose own public key is widely published – so all players are fairly sure of it. The certificate authority will (for a fee) perform a service to a player (say Alice) by verifying her identity (say, in person), and then issuing her a digitally signed document (using the CA's key) that testifies that the public key "P" belongs to the person Alice, who lives at 1200 East California Boulevard, Pasadena, California 91125, is a Caltech freshman, etc.

How can this service be used to defeat the vulnerability of the Diffie-Hellman protocol to a man in the middle attack?

6. **(**10 **points)** Show that cracking the El Gamal encryption algorithm and the Diffie-Hellman key exchange protocol, are computationally "equivalent", in the sense that a polynomial-time algorithm for either task implies a polynomial-time algorithm for the other.

A. **Open Problem.** The "Diffie-Hellman problem" is this: given a prime p, a primitive element g of Z/p, and $g^a \bmod p$ and $g^b \bmod p$, *compute* $g^{ab} \bmod p$. Obviously, this is the task which needs to be accomplished in order to crack the Diffie-Hellman key exchange protocol; and obviously, also, it is no harder than the discrete logarithm problem. However, no one knows whether the discrete logarithm problem reduces to the Diffie-Hellman problem.

B. **Open Problem.** Compute (in probabilistic polynomial time) a generator g of $(Z/p)*$, given a prime p.

The fact that this problem is open doesn't preclude use of discrete-log-based cryptosystems. There is a method of jointly producing a pair (p, g), that doesn't work by first producing p and then looking for g.