Written problems

1. Let m be a positive number.

   (a) Prove that if $a, b \in (\mathbb{Z}/m\mathbb{Z})^\times$, then $ab\%m \in (\mathbb{Z}/m\mathbb{Z})^\times$. (The unit group is "closed under multiplication.")

   (b) Prove, by induction on n, that if $a \in (\mathbb{Z}/m\mathbb{Z})^\times$, then $a^n\%m \in (\mathbb{Z}/m\mathbb{Z})^\times$ for all nonnegative integers n.

   (c) Prove, using (b), that if $a \in (\mathbb{Z}/m\mathbb{Z})^\times$, then $a^n\%m \in (\mathbb{Z}/m\mathbb{Z})^\times$ for all *integers* n (positive, negative, or zero). Remember that negative powers in modular arithmetic are defined in terms of the modular inverse.

2. Textbook exercise 2.8. Use a computer for the arithmetic. For part (d), I suggest using your naive discrete logarithm function from Problem Set 2. (Elgamal examples)

3. Suppose that m and n are integers such that $\gcd(m, n) = 1$.

   (a) Prove that if $a \in \mathbb{Z}$ is divisible by both m and n, then $mn \mid a$. (*Hint:* use Euclid's lemma).

   (b) Suppose that $a, b \in \mathbb{Z}$ satisfy the two congruences

   $$\begin{aligned} a &\equiv b \pmod{m} \\ a &\equiv b \pmod{n}. \end{aligned}$$

   Prove that $a \equiv b \pmod{mn}$ as well.

4. Textbook exercise 2.10, parts (a), (b), and (c). (On a three-transmission cryptosystem)

5. Write a function that reduces the problem of breaking the cryptosystem in the previous problem to the Diffie-Hellman problem. That is, assumed that you have an efficient function `dhOracle(p,g,A,B)` that extracts the shared secret from the public parameters and transmitted values in Diffie-Hellman, and use it to write a function `analyze210(p,u,v,w)` that would efficiently find the plaintext m in the system from exercise 2.10. It is fine to write the code by hand. (Obviously I cannot autograde it because I am unwilling to confirm or deny that I have a Diffie-Hellman oracle at this time.)

   Hint. The reduction is a little tricky to find; think about all the different ways you could match up the information you know with the g, A, B from Diffie-Hellman.

Programming problems

1. Write a program which deciphers a message sent to you with Elgamal encryption, given the public parameters p, g, your private key a, and the ciphertext (c_1, c_2). More specifically, write a function `decipherElgamal(p,g,a,c1,c2)` that returns the plaintext `m` (notation as in the table on p. 72). The length of the prime will vary, up to as large as 256 bits.

2. In the ElGamal cryptosystem, it is crucial that Bob generates a new random element (ephemeral key) k for every transmission, for reasons we will discuss in class. In this problem, you will implement some code that Eve might use to break Bob's encryption if he is not careful. Specifically, suppose that Bob sends two messages using the same value of k, and that Eve

somehow learns the contents of the first message (we discuss in class some reasons why this is not implausible).

Specifically, suppose that all parties know the paramters g and p and Alice's public key A, and Eve intercepts two transmissions from Bob to Alice. The first is a ciphertext `(c11, c12)`, which Eve determines corresponds to a plaintext `m1` (perhaps a standard greeting, or a weather report). The second is a ciphertext `c21,c22` corresponding to a plantext `m2` that Eve hopes to extract. Write a function `analyzeElgamal(g,p,A,c11,c12,m1,c21,c22)` that returns `m2`, assuming that Bob has foolishly used the same value of k for both transmissions.

The largest test cases will use 256-bit primes p, but a naive appraoche will earn partial credit.

3. Alice and Bob use ElGamal encryption on a regular basis, using public parameters p, g. Alice's public key is A. Eve has intercepted two ciphertexts `(c11,c12), (c21,c22)` from Bob to Alice, and has determined in some way that that the plaintext of the first transmission is a specific number `m1` (e.g. it is a standard greeting).

   Furthermore, Eve has a hunch that Bob is not generating his ephemeral keys very well. After last week's problem set, he knows better than to use the same ephemeral key twice, but the keys are still related. In particular, if `k1` was Bob's ephemeral key for the first transmission, his ephemeral key for the second was computed in this way:

   ```
   k2 = u * k1 + v
   ```

   where `u,v` are two integers between 1 and 100 that Eve does not know.

   Write a function `relatedkeys(g,p,A,c11,c12,m1,c21,c22)` that Eve could use to compute the second plaintext `m2` under these assumptions. The function should return a single integer, the value `m2`. The largest test cases will use 256-bit primes p.