Note. Problem Set 10 is be longer than usual since there was no problem set due the week of the exam, and is not due until **Friday** 13 May.

Written problems:

1. Samantha and Victor agree to the following digital signature scheme. The public parameters and key creation are identical to those of ECDSA. The verification procedure is different: to decide whether (s_1, s_2) is a valid signature for a document d, Victor computes

$$\begin{aligned} w_1 &\equiv s_1^{-1}d \pmod q \\ w_2 &\equiv s_1^{-1}s_2 \pmod q, \end{aligned}$$

then he checks to see whether or not

$$x(w_1G \oplus w_2V)\%q = s_1.$$

If so, he regards (s_1, s_2) as a valid signature for d.

Determine a signing procedure for this signature scheme, i.e. a procedure that Samantha can follow to produce a valid signature on a specific document. Your procedure should be non-deterministic, like in ECDSA.

2. Consider the following variant of EC Diffie-Hellman key exhange, in which Alice and Bob only exchange individual numbers, rather than both coordinates of a point on an elliptic curve.

- **Public parameter creation:** same as in table 6.5.
- **Private computations:** same as in table 6.5.
- **Public exchange of values:** Alice sends *the x-coordinate* of Q_A to Bob; Bob sends *the x-coordinate* of Q_B to Alice.
- **Further private computations**: Both Alice and Bob determine the *x-coordinate* of $(n_A \cdot n_B)P$. This is their shared secret value.

(a) Prove that if Q, Q' are two points on an elliptic curve with the same x-coordinate, and n is any integer, then nQ and nQ' also have the same x-coordinate.

(b) Describe how Alice is able to (efficiently) determine the shared secret, using only the information that she knows. You may assume that Alice has an efficient algorithm to determine square roots modulo p.

(c) What advantages, if any, does this system have over the usual ECDH system described in table 6.5?

3. Textbook exercise 7.1 (6.1 in 1st edition) ("1TRU" examples)

4. Suppose that the "1TRU" system (the system from the first section of §6 in the first edtion and §7.1 of the second edition) is modified as follows.

- Four parameters F, G, M, R are chosen at the beginning, in addition to the modulus q.
- When Alice makes her private and public keys, she chooses f, g so that $0 \leq f < F$ and $M \leq g < G$. She then computes her public key h as before.

- When Bob sends a message, he must choose his message m so that $0 \le m < M$, and he must choose his random number r so that $0 \le r < R$. Then he computes e as before.

(a) What are the values of F, G, M, R (in terms of q) used in the original version of "1TRU?"

(b) Suppose Alice computes b from e as in "1TRU." Write an inequality in terms of F, G, M, R, and q that will guarantee that the number b she computes is definitely equal to m.

(c) Explain why decryption could fail if the requirement $M \le g$ were dropped in Alice's procedure for generating her public key.

5. Suppose that Alice and Bob are using NTRU with $N = 251$, $q = 131$, $p = 3$, and $d = 6$.

(a) How many possible plaintexts are there? How many possible ciphertexts are there?

(b) Suppose every possible B_1-bit number is encoded in some way as an NTRU plaintext (with these parameters). How large can B_1 be for this to be possible? You don't need to describe an actual way to do this encoding, but I encourage you to think about how it can be done! **In all parts of this problem, you do not have to worry about rounding up or rounding down to an integer; for example, feel free to express your answer in terms of a logarithm. This simplifies the answers somewhat.**

(c) Suppose in turn that every cipher text is encoded as a B_2-bit number. How large must B_2 be in order for this to be possible?

(d) The *message expansion ratio* is the ratio B_2/B_1, where B_1, B_2 are as above. What is the message expansion ratio for NTRU with these parameters?

(e) Similarly, count the number of possible plaintext, and number of possible ciphertexts, for RSA encryption and Elgamal encryption. What are the message expansion ratios for those systems?

6. In some implementations of NTRU, rather than fixing one public parameter d, one chooses three different parameters d_1, d_2, d_3, and stipulates that Alice chooses $\mathbf{f}$ from $\mathcal{T}(d_1+1, d_1)$ and $\mathbf{g}$ from $\mathcal{T}(d_2, d_2)$, and Bob chooses $\mathbf{r}$ from $\mathcal{T}(d_3, d_3)$. Assuming that $d_1 \ge d_2 \ge d_3$, determine an inequality of the form $q > \cdots$ to replace the inequality $q > (6d+1)p$ in table 7.3, serving the same purpose in this more general formulation (your inequality should specialize to $q > (6d+1)p$ in the case $d_1 = d_2 = d_3 = d$).

Programming problems:

1. Write a function `ecDLP(P,Q,A,B,p,q)` to solve the elliptic curve discrete logarithm problem, in cases where the prime p is up to 28 bits. Here, P, Q are points on the curve $Y^2 \equiv X^3 + AX + B \pmod{p}$, and you are given, for convenience, the number q of points on the curve (which you may assume to be prime). The function should return the minimum nonnegative n such that $n \cdot P = Q$. Note that if you find any such solution n', then you can find the minimum solution by computing $n' \pmod{q}$ (we will discuss why this is true in class soon).

   A naive trial-and-error approach will earn partial credit, but to solve all test cases I recommend adapting the BSGS algorithm to the setting of elliptic curves. Note: one issue you may encounter is that it is not possible to place a "list" (e.g. `[2,3]`) into a Python `dict`. To resolve this, make sure all of your elliptic curve points (besides $\mathcal{O}$) are represented as "tuples" instead (e.g. `(2,3)`, with parentheses instead of brackets).

2. Write a function `ecdsaVerify(V,d,s1,s2)` that determines whether (s_1, s_2) is a valid ECDSA signature for the document d and the public (verification) key V (see p. 461 in the 1st edition or p. 322 in the 2nd edition for notation regarding ECDSA), using Elliptic curve "P-384." You should look up the specifics of curve P-384, e.g. by finding the relevant information in the standards document at the link below. You will also need to look up how to convert hexadecimal strings to integers in Python.

   `https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf`

   You may enjoy looking through the standards document to see what other sorts of information it contains.

3. In this problem, you will write several functions to work with "convolution polynomials," i.e. the set we have been denoting in class by R. These functions will be needed to implement NTRU decryption. All five functions take an argument N, the number of coefficients in each polynomial. For all five functions, elements of R will be represented in Python as a list of N integers; `f[i]` is the X^i coefficient of the polynomial. You will write all five functions in the same file; the test bank will test all of them.

   The five functions are:

   - `addR(f, g, N)`, which should return $f(x) + g(x)$ (as a list of N integers).
   - `scaleR(c, f, N)`, which takes an integer c, polynomial $f(x) \in R$, and returns the product $c \cdot f(x)$ (scale all coefficients by c).
   - `convolveR(f, g, N)`, which takes two polynomials $f(x), g(x) \in R$, and returns $f \star g$.
   - `reduceR(f, p, N)`, which takes $f(x) \in R$ and returns the result of reducing f modulo p.
   - `centerliftR(f, p, N)`, which takes $f(x) \in R$ and returns the result of "centerlifting" f modulo p (we discuss this concept on Friday).

4. Write a function `decipherNTRU(N, p, q, d, h, f, g, Fp, Fq, e)` that deciphers an NTRU ciphertext, given all of Alice's private information. See the NTRU summary table (p. 394 in 1st edition; p. 419 in 2nd edition) for the notation (unfortunately, there's a lot of it!). In input, all elements of R will be "centerlifted," so they may have both positive and negative coefficients. You should return the plaintext `m` centerlifted modulo p.

   Note that you may discover that you do not need all ten of these arguments in order to compute the result. I have included all of them for completeness. It's also worth pointing out that in principle I don't need to tell you $\mathbf{F}_p$ and $\mathbf{F}_q$, since they can both be computed from $\mathbf{f}$, but I am not expecting you to implement the algorithms needed to do this.

5. **This problem is now extra credit, since we may not cover the relevant content until Wednesday 5/11.**

   You will be given the public parameter q, Alice's public key h, and a ciphertext e for the congruential cryptosystem (which I called "1TRU" in class). Write a function `break1TRU(q, h, e)` that extracts the plaintext (without knowing the private key).

   Half of the test cases will be small enough that a brute force approach should work. For the remaining test cases I suggest that you use Gauss's lattice basis reduction algorithm.