Final Exam

Due Thursday, December 17 **at noon**

This is the final assignment of the course. As in the takehome projects:

- Your work must be your own. For this assignment do not work in groups or share code.
- It is open book: you may use the textbook, your class notes, my class notes and lecture videos, past homework assignments and their solutions, as well as the Sage and Python documentation. Everything else is off limits. I can also be a resource so don't hesitate to reach out or come by office hours!

If you have questions, please reach out to me ASAP.

Implementation Part

1. Implement Lenstra's elliptic curve factorization algorithm. In particular, write function `LenstraFactor(N)` which takes as input a number N and returns a nontrivial factor of N. The algorithm is outlined in the December 1st lecture (or Table 6.8 in [HPS]). You will need your elliptic curve arithmetic algorithms from Takehome 2, including `addPoints` and one of your variants of `doubleAndAdd`. You will likely need to adjust them to deal with the case where adding is impossible (this is, after all, what you are looking for).

2. This is the third integer factorization algorithm we have implemented (there are many more). In particular, we have seen `PollardFactor` (HW6 Problem 1) which implements Pollard's $p-1$ algorithm, and `PollardRhoFactor` (HW10 Problem 3) which implements Pollard's ρ method. Let's compare them here. Each of the following numbers are pq for distinct primes p and q. Run (or try to run) each factorization algorithm on each.

   (a) 25992521

   (b) 70711569293

   (c) 508643544315682693

   (d) 25377042799063401776035673 83

   Remark. *Not every algorithm will terminate with all four numbers above (although I found each number to be able to be factored by at least one of them), so it may be pertinent to put some upper bounds in your looping. I found an upper bound for j of 10 or 20 thousand for Lenstra's algorithm to be useful, although you could go higher (definitely not lower). I think it is also a good idea to try several curves before giving up. An upper bound of 100,000 for Pollard's $p-1$ algorithm seems reasonable, and 1,000,000 for Pollards ρ (with a possibility of trying several mixing functions) also seems to hit the sweetspot for me. But feel free to play with these numbers in trying to find your factorizations.*

Written Part

3. In Problem 2 I found one algorithm worked for (c) while a different one worked for (d). Explain why this could happen.

4. In class we suggested that the problem of factoring a number N is in some sense equivalent to being able to compute square roots modulo N. In this problem we will make this precise, for $N = pq$ a product of 2 distinct **odd** primes.

   (a) Suppose you know the factorization of N into pq. Describe an algorithm to efficiently compute whether a has a square root modulo N, and prove the correctness of your algorithm.

   (b) Suppose $\gcd(a, N) = 1$. Show that if a has one square root modulo N, then it exactly 4 square roots modulo N. In the case where $\gcd(a, N) \neq 1$, how many square roots might a have? Why?

   (c) Suppose you know the factorization of N into pq. Describe an algorithm to compute all the square roots of a modulo N if they exist. Prove the correctness of your algoritm. (You may assume you have a fast algorithm to compute square roots modulo primes.)

   (d) Conversely, suppose you have an oracle that can tell you all the square roots of a modulo N if they exist. Describe a way to use constultation with this oracle to factor N. Prove your method works.

5. Let E be an elliptic curve over $\mathbb{F}_p$, and suppose that $\#E(\mathbb{F}_p) = n$. Let $P \in E(\mathbb{F}_p)$ be a point. Suppose a is an integer and that $\gcd(a, n) = 1$. Prove that P is a multiple of aP.

6. Recall that the Elliptic Curve Diffie Hellman Problem (ECDHP) is the problem of recovring a point $nmQ \in E(\mathbb{F}_p)$, given nQ and mQ.

   (a) Formulate the MV-Elgamal Problem. That is, describe the precise mathematical problem at the center of MV-Elgamal.

   (b) Prove that the security MV-Elgamal Problem is equivalent to the security of the ECDHP. (Use oracles!).

7. Recall that in HW4 Problem 3 we studied a public key cryptosystem that involving Alice and Bob agreeing on a public prime and exchanging a series of values in $\mathbb{F}_p^*$.

   (a) Formulate a version of this cryptosystem for Elliptic curves, and prove its correctness. You may assume that you have a fast way to compute $\#E(\mathbb{F}_p)$. Be sure to make clear what is public information and what is private information. What plays the roll of the message?

   (b) Show that a solution to the Elliptic Curve Discrete Log Problem will break this cryptosystem.

   (c) Show that a solution the the Elliptic Curve Diffie Hellman Problem will break this cryptosystem.

   (d) Discuss any other advantages or disadvantages you observe about this cryptosystem.

   In the previous problem, the *message* which was sent was a chosen point on an elliptic curve. When the message is a number we can use ASCII to directly translate text to integers, but with points on an elliptic curve, the question is more subtle. MV-Elgamal got around this subtlety just using the elliptic curve to create a shared secret, and using the coordinates as keys for a symmetric cipher whose plaintext are integers (not points). In the following exercise we will describe another solution, using a probabilistic algorithm to map integers to points on the elliptic curve in a recoverable way.

8. Let E be an elliptic curve over $\mathbb{F}_p$ with equation $y^2 = x^3 + Ax + B$, where p is $2k+1$-bits in length.

   (a) The most naïve way to map a plaintext m (which can be thought of as an integer) to a point on the elliptic curve would to embed m as the x-coordinate. Explain what goes wrong with this approach.

   (b) Instead we will implement the following probabilistic algorithm, which will allow us to map a message k-bits in length to a point on the elliptic curve, in the following steps.

   (1) Start with a plaintext message m, stored as an integer k-bits in length.
   (2) Choose a random integer r, also k-bits in length.
   (3) Compute $r||m \in \mathbb{F}_p$.
   (4) Detect if $r||m$ is the x coordinate of a point in $E(\mathbb{F}_p)$. If so, compute the y coordinate, and return $P = (r||m, y)$ as your plaintext for the elliptic curve encryption. Otherwise return to step 2.

   Describe exactly how Step 4 would be carried out. (You may assume that you have a fast algorithm to compute square roots modulo p if they exist). Conversely explain how to reverse the algorithm to recover the plaintext from a point.

   (c) How many values of r would expect to have to try until you have a point? Justify your answer. (Hint, this is essentially a collision algorithm, so what is the length of the list you are trying to find a collision with? You may assume $r||m$ varies 'randomly enough' as r varies.)

   (d) Explicitly describe the steps you need to add to the algorithm from Problem 7 in order to be able to communicate in plain language. (Including applying things like `intToText`).

9. You are Eve, and you intercept Bob's cipher text $c = [R, c_1, c_2]$ sent to Alice. You know it was encrypted using MV-Elgamal, with a public curve E over a prime p with a base point P, and you know Alice's public key Q. You also have 1 time access to Alice's machine to decrypt MV-Elgamal ciphers. Describe a way to obtain the plaintext message Bob sent to Alice, proving the correctness of your method. **Note:** Alice's machine runs a security protocol which will not decrypt messages that have already been decrypted, so you can not just ask it to decrypt c.

   Bonus: Alice noticed a weakness in her machine's security protocol, and updates it to the following: If Alice's machine has already decrypted $c = [R, c_1, c_2]$, it will only decrypt ciphertexts $c' = [R', c_1', c_2']$ where $c_1' \neq c_1$, $c_2' \neq c_2$, and $R' \neq \pm R$. But you're clever: devise a 'chosen plaintext attack' to get around this updated security protocol.