Final Assignment

Due Thursday, December 16 **at 11:00 PM**

This is the final assignment of the course. As in the 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!

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 November 30th lecture (or Table 6.8 in [HPS]). You will need your elliptic curve arithmetic algorithms from HW10 and Project 2, including `addPoints`, `doubleAndAddSmall`, and `generateEllipticCurveAndPoint`. You will likely need to make adjustments 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 `sieveFactor` (HW7 Problem 3) which implements the Pomerance's quadratic sieve. 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. **In the written part, include a section commenting on the results. Why did different algorithms have more success with the some numbers and not others?**

   (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 you may want to experiment putting upper bounds in your looping. For* `LenstraFactor`*, I think it is also a good idea to try several curves before giving up.*

Written Part

3. 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.

4. Recall that in HW4 Problem 6 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. We need to introduce the following notation: if x, y are 2 bitstrings, then $x||y$ is their *concatenation*, that is, the bit string which first consists of x, and then y. For example $01011||001 = 01011001$.

5. 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? (You may assume that $r||m \in \mathbb{F}_p$ varies randomly as r does, and use the black box about expected values given in Project 1 6(d)).

The algorithm described in the previous problem has a few downsides. First, there's some randomness which slows down the encryption process. Second, one has to use primes much larger than the

messages you start with, leading to significantly increased message expansion. In practice, one instead can choose special elliptic curves where one can define a bijection between $\mathbb{F}_p$ and $E(\mathbb{F}_p)\setminus\{\mathcal{O}\}$ (these are called *supersingular* elliptic curves). Let's explore an example of how this might work.

6. For this entire problem we fix a prime $p \equiv 2 \mod 3$, and consider the elliptic curve given E by the equation $y^2 = x^3 + 1$ over $\mathbb{F}_p$.

   (a) Show that every $y_0 \in \mathbb{F}_p$ is the y coordinate of exactly one nonzero point of $E(\mathbb{F}_p)$. (*Hint:* use HW8 Problem 6(e)).

   (b) To implement this, we have to make HW8 Problem 6(e) explicit: let $z \in \mathbb{F}_p$. Show that $z^{\frac{2p-1}{3}}$ is the unique cube root of z. (Don't forget to show that $\frac{2p-1}{3}$ is an integer).

   (c) With part (a) and (b) in mind, write algorithms `textToPoint` and `pointToText` which will translate a string to a point on E, and vice versa. (You may call the functions `intToText` and `textToInt` in your pseudocode.)

7. 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 their 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 ciphertext attack' to get around this updated security protocol. (You may assume you have a fast algorithm to compute $\#E(\mathbb{F}_p)$.