Takehome Project 2

Due Saturday, December 5

In this assignment you will implement from the ground up a variant of Elliptic Curve Elgamal (callled Menezes-Vansone Elgamal or MV-Elgamal for short). This includes implementing all the basics for elliptic curve arithmetic, including addition and 'fast powering.' You will also prove a few things about the algorithms you write. As in the first project:

- 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!
- Part of the assignment is involves me sending a message to you and you replying. Therefore you must share an Elliptic Curve Elgamal public key with me (on Discord or via email if you prefer), **by Thursday!**

If you have questions, please reach out to me ASAP. Ok, let's get started. Have fun!

Implementation Part

0. First gather your belongings. You will need the extended euclidean algorithm, your implementations of the ASCII encoding schemes: `textToInt` and `intToText`, `getBinary` from Homework 2, as well as `findPrime` and all its dependencies from the first project.

1. Let's start by setting up and testing the basic data structures for our objects. For our purposes we will save an elliptic curve as a pair of integers: `E = [A,B]`, corresponding to the equation $y^2 = x^3 + Ax + B$. For this assignment, all elliptic curves will be over a finite field $\mathbb{F}_p$. A point of an elliptic curve will either be an ordered pair $(x, y) \in \mathbb{F}_p^2$, or else the point $\mathcal{O}$ at infinity. We will save a point in $\mathbb{F}_p^2$ as a ordered pair `P = [x,y]`, and the point at infinity as a character `'O'`.

   (a) Recall that an equation $y^2 = x^3 + Ax + B$ gives an elliptic curve precisely when the discrimant $\Delta = 4A^3 + 27B^2$ is nonzero. Write a function `isElliptic(E,p)` which takes the equation of a (potential) elliptic curve and a prime p, and returns `True` if it is in fact an elliptic curve (mod p), and `False` otherwise.

   (b) Recall that $E(\mathbb{F}_p)$ is the set of points $(x, y) \in \mathbb{F}_p^2$ satisfying the equation for E, together with a point $\mathcal{O}$ at infinity. Write a function `onCurve(P,E,p)` which takes as input a point P, an elliptic curve E, and a prime p, returning `True` if $P \in E(\mathbb{F}_p)$ and `False` otherwise. (Be sure to include the possibility that $P = \mathcal{O}$).

   (c) Consider the elliptic curve E given by $y^2 = x^3 + 3x + 2$ and the point $P = (3, 5)$. Using the algorithms above: for the first 7 odd primes $(3, 5, 7, 11, 13, 17, 19)$ print the answer the following questions.

      i. Is E an elliptic curve over $\mathbb{F}_p$?

      ii. Is $P \in E(\mathbb{F}_p)$?

      iii. Is $\mathcal{O} \in E(\mathbb{F}_p)$?

   (d) Print the list of points for $y^2 = x^3 + 3x + 2$ over $\mathbb{F}_7$.

2. Now let's throw some arithmetic into the mix.

   (a) Write a function `addPoints(P,Q,E,p)`. It should take as input an elliptic curve E and a prime p, together with two points $P, Q \in E(\mathbb{F}_p)$, and should return the sum $P + Q \in E(\mathbb{F}_p)$. Use the elliptic curve addition algorithm described in class. (Be sure to allow for P, Q, or $P + Q$ to be $\mathcal{O}$).

   (b) Let's do some testing:

      i. Let E be $x^3 + 3x + 8$, over $\mathbb{F}_{13}$, $P = (9, 7)$ and $Q = (1, 8)$. Compute $P + Q$, $2P$ and $\mathcal{O} + Q$. (Note: the entire multiplication table for this curve is worked out in [HPS] Table 6.1, so it would be a good example to troubleshoot with).

      ii. Let E be $y^2 = x^3 + 3x + 2$ over $\mathbb{F}_7$. Print the entire multiplication table for E. (You can avoid adding a new line at the end of a print command as follows: `print(stuffToPrint,end='')`, and print $\mathcal{O}$ as `[0000]` to keep rows even).

      iii. Let E be $y^2 = x^3 + 231x + 473$ over $p = 17389$. Let $P = (11259, 11278)$ and $Q = (11017, 14673)$. Compute $P + Q$, $2Q$ and $3P$.

3. Given $P, Q \in E(\mathbb{F}_p)$ such that $Q = nP$, the Elliptic Curve Discrete Log Problem (ECDLP) is the problem of finding n. The security of the ECDLP relies on the fact that computing adding P to itself n times is slow for large n. But in order to harness this, we need to be able to compute nP more quickly. For the DLP over $\mathbb{F}_p^*$, we used the *fast powering algorithm.* We will implement an analog of this here.

   (a) Write an algorithm called `doubleAndAdd(P,n,E,p)` which takes as input an elliptic curve E, a prime p, a positive integer n, and a point $P \in E(\mathbb{F}_p)$, returning $nP \in E(\mathbb{F}_p)$ using the double and algorithm from class:

      (1) Compute the binary expansion $[n_0, n_1, \cdots, n_r]$ of n.

      (2) Compute $Q_i = 2^i P \in E(\mathbb{F}_p)$ for $0 \leq i \leq r$ by successive doubling. (Notice that $Q_{i+1} = 2Q_i$)

      (3) Compute $nP = n_0Q_0 + n_1Q_1 + \cdots + n_rQ_r \in E(\mathbb{F}_p)$ in at most r additions.

   (b) Like in HW2, for large values of n `doubleAndAdd` requires a remembering large amounts of data (including $\log_2(n)$ different multiples of P), which is not ideal. Adapt HW 2 Problem 3(b) (`fastPowerSmall`) to the elliptic curve case. That is, write an aglorithm `doubleAndAddSmall(P,n,E,p)` with the same inputs, but returning nP with $\mathcal{O}(1)$ storage space. (In step (3) of HW3 3(b) you compute a floor, do this with the integer division operator: `//`).

   In class we saw that we could decrease the number of additions in step (3) of `doubleAndAdd`, at the expense of at most one doubling in step (2) if we used a certain ternary expansion of n instead of the binary expansion in step (1).

   (c) Write a function called `getTernary(n)` which return a ternary expansion of n (with coefficients in $\{-1, 0, 1\}$). It should first call `getBinary(n)` and then should follow the algorithm described at the end of class on November 19, or Proposition 6.18 in [HPS] to turn the binary expansion of n into a ternary expansion at least half of which consists of zeros.

   (d) Recall that this sort of approach works better for elliptic curves because one can find the inverse of a point quite quickly. Write a function called `invertPoint(P,p)` which takes a point P (presumed to be on an elliptic curver over $\mathbb{F}_p$), and returns $-P$ in $\mathcal{O}(1)$ basic operations (or even 1 basic operation!). Think about why E didn't have to be part of the data you sent the function. Also make sure the function also works for the point at infinity.

(e) Write an algorithm called `doubleAndAddTernary(P,n,E,p)`, with the same inputs and output as before. It shold look almost identical to the algorithm from 3(a), with adjustments to steps (1) and (3).

Notice that `doubleAndAddTernary` still needs to remember lots of data, including about $\log_2(n)$ different multiples of P. Since in principle we may need the entire binary expansion to compute the ternary one, we cannot directly condense the storage like in `doubleAndAddSmall`. Nevertheless, in practice $\log_2(n)$ binary digits is the same amount of storage it takes to store the number n, so remembering this many digits isn't the huge issue, it is the $\log_2(n)$ multiples of P (which can in practice each have coordinates a hundred digits long) that is the issue. Let's finish this exercise by removing the need to store all that.

(f) Write an algorithm called `doubleAndAddTernarySmall(P,n,E,p)` with the same inputs and output as before, adjusting the algorithm from part (e) to eliminate the need to store all the multiples of n. (Hint: to pass from `doubleAndAdd` to `doubleAndAddSmall` you had steps (1) through (3) of the algorithm run simultaneously in one loop. In this case you have to do step (1) ahead of time, but can still combine steps (2) and (3) into one loop.)

(g) Now let's test them out!

i. Let E be $y^2 = x^3 + 14x + 19$ over the prime $p = 3623$. Let $P = (6, 730)$. Compute $947P$ with all four algorithms above. (This example is worked out in detail in [HPS] Example 6.16 and Table 6.4, so this would be a good example to troubleshoot with).

ii. Let E be $y^2 = x^3 + 143x + 367$ over the prime $p = 613$. Let $P = (195, 9)$. Use all four algorithms to compute $23P$.

4. Many applications of elliptic curves (Including MV-Elgamal of problem 5) require as input an elliptic curve E together with a (nonzero) point $P \in E(\mathbb{F}_p)$. One way to do this would be to start with an elliptic curve $y^2 = x^3 + Ax + B$, choose a random value for x, compute the right side of the equation, and then compute a square root to find y. That being said, so far we only have an algorithm for square roots if $p \equiv 3 \mod 4$, and we don't have space within this assignment to describe the general algorithm. Instead, one can start with a point in $\mathbb{F}_p^2$ and a random value of A to derive B. (This method will also come in handy for Lenstra's Elliptic Curve factorization algorithm where we need a point in $E(\mathbb{Z}/N\mathbb{Z})$ where N is not necessarily prime, and square roots become more elusive).

(a) Write an algorithm `generateEllipticCurveAndPoint(p)` which takes as input a prime p and outputs an elliptic curve `E = [A,B]` and a nonzero point `P = [x,y]` on the elliptic curve. It should begin by randomly selecting a point $P \in \mathbb{F}_p^2$, and $A \in \mathbb{F}_p$, and should then derive B from that data using the equation $y^2 \equiv x^3 + Ax + B \mod p$. Make sure to check that the discriminant is nonzero (problem 1(a)) before returning the curve.

(b) Generate a curve and point over the following finite fields, confirming that the generated point lies on the curve using 1(b).

i. $\mathbb{F}_{13}$.

ii. $\mathbb{F}_{1999}$.

5. Now that everything is set up, we will implement the following variant of Elliptic Curve Elgamal, called Menezes-Vanstone Elgamal (MV-Elgamal). We summarize the process in the table below (this is also summarized in Table 6.13 of [HPS]).

Public Parameter Creation	
A large prime p, an elliptic curve E over $\mathbb{F}_p$, and a point $P \in E(\mathbb{F}_p)$ of large order is selected and published.	
Alice	**Bob**
Public Key Generation	
Choose secret multiplyer n_A Compute $Q_A = n_A P$ If $Q_A = \mathcal{O}$ or $(x_A, 0)$, choose a new n_A. Publish the public key Q_A.	
Message Encryption	
	Choose 2 plaintexts, $m_1, m_2 \in \mathbb{F}_p^*$. Choose random number k. Compute $R = kP, S = kQ_A \in E(\mathbb{F}_p)$. If either $R, S = \mathcal{O}$, choose new k. Write $S = (x_S, y_S)$. If $x_S = 0$ or $y_S = 0$, choose a new k. Compute $c_1 = x_S m_1 \mod p$. Compute $c_2 = y_S m_2 \mod p$. Send ciphertext (R, c_1, c_2) to Alice.
Message Decryption	
Compute $T = n_A R$ Write $T = (x_T, y_T)$ Compute $m_1' = x_T^{-1} c_1 \mod p$ Compute $m_2' = y_T^{-1} c_2 \mod p$ $(m_1', m_2') = (m_1, m_2)$ **is the message!**	

Let's implement the various peices of this.

(a) Write a function `MVParameterCreation(b)` which generates an elliptic curve E and a nonzero point $P \in E(\mathbb{F}_p)$ over a prime p of b bits, returning E, P, and p in an array `pubParams = [E,P,p]`. (Use problem 4 and `findPrime`. Make sure that P has order > 2, that is, that it's y-coordinate is nonzero).

(b) Write a function `MVKeyCreation(pubParams)` which takes as input public parameters as above (an array consisting of a prime p, an elliptic curve E over $\mathbb{F}_p$, and a point $p \in E(\mathbb{F}_p)$), and returns a pair `[privateKey,publicKey]` consisting an an MV-Elgamal private key (which is an integer) and an MV-Elgamal public key (which is a point on $E(\mathbb{F}_p)$).

(c) Write a function `MVEncrypt(pubParams,m1,m2,publicKey)` which takes as inputs the public parameters, 2 plaintexts m_1 and m_2 (as elements of $\mathbb{F}_p^*$), and Alice's MV-Elgamal public key (which is a point on $E(\mathbb{F}_p)$). It should output ciphertext `cipherText = [R,c1,c2]` where $R \in E(\mathbb{F}_p)$ and $c_1, c_2 \in \mathbb{F}_p^*$.

(d) Write a function `MVDecrypt(pubParams,cipherText,privateKey)` which should take as input the public parameters, MV-Elgamal ciphertext `[R,c1,c2]`, as well as Alice's private key (which is an integer). It should return a pair of plaintexts `[m1,m2]` which are elements of $\mathbb{F}_p^*$.

6. Now let's use this to communicate!

   (a) Generate a MV-Elgamal parameters and a key from a 32 bit prime. Use it to encrypt and then decrypt $m_1 = 314159$ and $m_2 = 8675309$. Did it return the correct message?

   (b) Let's use MV-Elgamal to communicate over a public channel. Generate public parameters and an MV-Elgamal key from a 512 bit prime. Post your public key on the Discord channel `MV_Keys` **by THURSDAY!**. **Save your private key!** Expect a message from me.

   (c) I have generated the following MV-Elgamal parameters and public key from a 512 bit prime. You will need this information to reply to my message. My prime number is:
   $p =$
   96679916709418467804792071758669421219132225040927948189628129001934696211369
   11908758877261892843473911159087629773888624493983717963602737226560545528587

   My elliptic curve is E given by $y^2 = x^3 + Ax + B$, where:
   $A =$
   48796847113108314817620167652038771298334458679335202235864999577957635352944
   10783255167400481968525677691282145732794658725633769637917211357465365360450
   $B =$
   26559398701073618928983075356828585238654740607301641675503015316472536393205
   65520039350930005199057213231234444992680757129682717380286803281748377447915
   On this is base point $P = (x_P, y_P)$ where:
   $x_Q =$
   89330529788603300242847785197359326655836846729382989053443059826783057432552
   13445242262118220952954107876124415331578156369096290711735309321059643838016
   $y_Q =$
   51641215831729412853911207774965535193045715932824763208358176281130260147025
   64367753036545237321623642430399030929755158264689239627151270468409553539588

   My public key is $Q = (x_Q, y_Q)$ where:
   $x_Q =$
   40017859370015171884208052549823682959627060028090512234481808049236030360129
   8845273961337755730657164420024682753462374059189559501229256175941740669988
   $y_Q =$
   48915195632784957833692175191157973184450301586242848481115270265647634791665
   76052473627513226258348242299697103325911991906892373233399304028909125087597

Written Part

7. Let's begin the written portion of this assignment by studying the correctness of the MV-Elgamal.

   (a) Let E be an elliptic curve modulo a prime p and $P \in E(\mathbb{F}_p)$ be a point not equal to $\mathcal{O}$. Prove that the order of P is 2 if and only if it's y coordinate is equal to 0. (This justifies the remark in 5(a)).

(b) Use part (a) to explain why in the key creation phase of MV-Elgamal it was insisted that Q_A must have a nonzero y coordinate.

(c) Prove the correctness of MV-Elgamal. (Be sure to explain why x_S, y_S must be nonzero).

(d) The original Elgamal and the Elliptic Curve Elgamal essentially wrap together a Diffie-Hellman key exchange and a symmetric cipher into one procedure. Explain why this is also the central philosophy of MV-Elgamal. What plays the roll of the shared secret? What about the symmetric key?

8. Let's now discuss the efficiency of MV-Elgamal

(a) What is the message expansion of MV-Elgamal?

(b) Suppose Alice has an algorithm to efficiently compute square roots modulo p (such things certainly exist). Explain a way that Bob could compress the ciphertext by sending 1 bit in place of the y-coordinate of R, and prove that your method works. What would the new message expansion be?

9. Let's conclude with some potential attacks on MV-Elgamal.

(a) Show that if Eve can solve the Elliptic Curve Discrete Log Problem she can use it to crack MV-Elgamal.

(b) Eve knows the elliptic curve E, and intercepted the two peices of ciphertext c_1, c_2. Explain how Eve could use this information to relate m_1 and m_2 in a polynomial equation (modulo p). In particular, if Eve knows one of the peices of plaintext, she can use that to recover the other peice by solving an equation modulo p. (The moral here is, both peices of plain text must be unique and private with each message. For example: Bob should never leave one blank just because he doesn't need the space.)