Takehome Project 2

Due Thursday, December 9

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 Tuesday December 7!**

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 3, as well as `findPrime` and all its dependencies from the first project. You will also need your implementation of elliptic curves and elliptic curve arithmetic−`isElliptic`, `onCurve`, and `addPoints`−from HW 10.

1. 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: `//`).

   (c) 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 both 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 both algorithms to compute $23P$.

2. Many applications of elliptic curves (Including MV-Elgamal below) 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 you actually get an elliptic curve.

   (b) Generate a curve and point over the following finite fields, and confirm that the point actually lies on the elliptic curve.

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

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

3. 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 Table 3 on the next page (this is also summarized in Table 6.13 of [HPS]). 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 2 and `findPrime`. Make sure that P has order > 2.).

   (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 $\mathbb{F}_p^*$.

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!**	

Table 1: MV-Elgamal Encryption Scheme.

4. 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 Tuesday 12/7!** *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:

   $p =$
   92383392100858213843708486493099510442875968361228244333693822706907049111155
   08332176060210977135155985773831184797326834780367925020889055994468078443941
   My elliptic curve is E given by $y^2 = x^3 + Ax + B$, where:
   $A =$
   33605871384886345486243343645671413529752464064739214288226273355810239309164
   03411066084961421298477347509247243673102565104835387508751726704037996443665
   $B =$
   52744656800731986534364264096568563057081218124763591754656897386132507062915
   77435278096171212292781654086205248915051174084842023901074497831143355053310
   On this is base point $P = (x_P, y_P)$ where:
   $x_P =$
   38948213259008292266615145735498357291258722225212867602196075351551404755663
   80261638014526024207181589942225595395557392022644429888469190609469251687317
   $y_P =$
   20103263670046222847700201327547333244062890046222369403286709444648611214641
   89430879199099400468690110164705702449330074512245604434419522655885266701508

   My public key is $Q = (x_Q, y_Q)$ where:
   $x_Q =$
   82853339218583582727934935199067725301905843196388566436577723897362821526762
   35947510470165952693773814539643126521222828014548593322716383090625534896280
   $y_Q =$
   53165305989086488568892235365210128737644237438730469630256050559211227031328
   45908355104500445027783863737649595841877172902771660618673858270962273557420

Written Part

5. 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.

   (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?

6. Let's now discuss the efficiency.

   (a) Compute the time complexity of the following algorithms (in terms of the number of bits of your prime), proving your answer is correct. You may assume the basic operations $+, -, \times, \div$ and choosing a random number are all $\mathcal{O}(1)$.

      i. `doubleAndAddSmall`
      ii. `generateEllipticCurveAndPoint`
      iii. `MVParameterCreation` (Recall you computed the time complexity of `findPrime` in Project 1
      iv. `MVKeyCreation`
      v. `MVEncrypt`
      vi. `MVDecrypt`

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

   (c) 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?

7. 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) Show that if Eve can solve the Elliptic Curve Diffie Hellman Problem, she can use it to crack MV-Elgamal

   (c) 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.)