Final Assignment: Written Solutions

Written Part

We begin with a brief comment on problem 2 from the implementation part. This is rather open ended, and was just a chance to encourage you to think a little bit about how various algorithms interact.

Proof. For me, in part (c) the $p-1$ algorithm didn't work, whereas in part (d) it did. This may be initially surprising but came down to the fact that one for one the prime factors p of the number from part (d), $p-1$ has small factors (which doesn't happen for the number in part (c)). On the other hand, Lenstra worked very quickly for (c) but in (d) I had to cycle through many different curves until I found one that worked almost instantly. This is because we eventually found an elliptic curve E together with a point P which had smooth order. This suggest that hopping around between curves is really the best way to get LestraFactor to work.

The quadratic sieve was a mess for me. It crashed my kernel for both (c) and (d). I think this is because we're just generating a HUGE matrix, and CoCalc doesn't want to allocate us that much memory. Clever use of data structures and more efficient linear algebra should in theory make it work (as the quadratic sieve should be the fastest), so something to keep in mind. □

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.

   Proof. By Lagranges theorem, we know that $nP = \mathcal{O}$. By the extended Euclidean algorithm, there exists some u, v such that $au = 1 + vn$. Therefore we may compute:

   $$u(aP) = (1 + vn)P = 1P + v(nP) = P + v\mathcal{O} = P,$$

   and so 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?

   Proof. We will formulate it in a table. Public information will be highlighted in purple. To begin, Even and Bob agree on the following public parameters: a prime number p, and an elliptic curve E over $\mathbb{F}_p$. The size $n = \#E(\mathbb{F}_p)$ is easily computed and can also therefore be assumed to be public. The message will be a point in this elliptic curve.

	Alice	Eve	Bob
1.	Alice has message $M \in E(\mathbb{F}_p)$		
2.	Alice chooses a with $\gcd(a, n) = 1$		
3.	Alice computes $P = aM \in E(\mathbb{F}_p)$	$\longrightarrow$	Bob recieves P
4.			Bob chooses b with $\gcd(b, n) = 1$.
5.	Alice recieves Q	$\longleftarrow$	Bob Computes $Q = bP \in E(\mathbb{F}_p)$
6.	Alice computes $a' = a^{-1} \mod n$		
7.	Alice computes $R = a'Q \in E(\mathbb{F}_p)$	$\longrightarrow$	Bob recieves R.
8.			Bob computes $b' = b^{-1} \mod n$
9.			Bob computes $S = b'R \in E(\mathbb{F}_p)$. Then $S = M$!

To prove correctness we must show that $S = M$. We first fix some notation $aa' = 1 + kn$ and $bb' = 1 + ln$. Then

$$aa'bb' = 1 + kln + (l^2 + k^2)n^2 = 1 + \lambda n$$

for some integer $\lambda = kl + (l^2 + k^2)n$. The we remind the reader that (as in problem 3), by Lagrange's theorem, $n \cdot T = \mathcal{O}$ for any $T \in E(\mathbb{F}_p)$. Then we can directly compute:

$$S = b'a'baM = (1 + \lambda n)M = M + \lambda nM = M + \lambda\mathcal{O} = M,$$

as desired. □

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

Proof. Eve intercepts P and $Q = bP$, and consults with the ECDLP oracle to recover b. She can then compute $b' = b^{-1} \mod n$ (as n is public). She then intercepts R and can compute $S = b'R$. But this is M, so Eve has discovered the message. □

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

Proof. Eve intercepts $P = aM$, $Q = abM$, and $R = aba'M$. She notices that $R = a'Q$ and $P = b'Q$. To see the latter, we directly compute

$$b'Q = abb'M = a(1 + ln)M = a(M + lnM) = a(M + \mathcal{O}) = aM = P.$$

Therefore she can consult an ECDHP oracle, feeding them, $Q, a'Q$, and $b'Q$ (which we have seen are all public), and obtain $a'b'Q$. But this is $a'b'abM = S = M$. □

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

Proof. This is rather open ended. One thing I notice, is the fact that they have to communicate 3 times to send a message, and that Bob must be available to respond immediately to receive a message. This also allows extra opportunity for security breaches, for example, Eve could pose as Bob since the public key isn't published, and relies on Alices message to be available.

Also there is the problem of turning a plaintext (in plain language) into a point on an elliptic curve (although we study this in the next two problems).

Then there is message expansion. Suppose p is k-bits. Since $\#E(\mathbb{F}_p) \approx p$, one can only store approximately k bits of data in the curve. But we have to send 2 coordinates in $\mathbb{F}_p$ back and forth 3 times, so this is a 6:1 message expansion. This could be compressed to 3:1 using the technique of TH2 Problem 5(d), but is still greater than many other systems.

This has an important advantage over the analogous problem over $\mathbb{F}_p^*$ described in HW4. In particular, the ECDLP is more secure than the DLP. □

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.

   Proof. We denote the right hand side of the equation of the curve by the function $f(x) = x^3 + Ax + B$. Give some value m, it isn't always true that $f(m) = m^3 + Am + B$ is a square. If this happens, m couldn't serve as an x-coordinate of any point in $E(\mathbb{F}_p)$. □

   (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.

   Proof. Let $z = r||m \in \mathbb{F}_p$, and consider $f(z) = z^3 + Az + B$. We first compute $\left(\frac{f(z)}{p}\right) = f(z)^{\frac{p-1}{2}} \bmod p$. If this is 1, then we compute a square root $y = \sqrt{f(z)}$

mod p, and our message will be the point $P = (z, y)$. (Notice if $\left(\frac{f(z)}{p}\right) = 0$ then we could compute a square root, but it would be 0. This would give us an order 2 point on the curve and not be very secure for communication. In general, we could check that P has relatively high order before continuing.)

To reverse this process, given a point $P = (x, y)$ we can recover the message m as the last k-bits of x. □

(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)).

Proof. We first recall from HW2 Problem 8(d) that exactly half of the elements of $\mathbb{F}_p^*$ are quadratic residues. Therefore we may assume that $r||m$ has a probability of $\rho = 1/2$ of having a square root. As in Project 1 6(d), we may therefore expect it to take $1/\rho = 2$ guesses of r to obtain a perfect square. □

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)).

Proof. Plug y_0 into the equation for the elliptic curve, and then solve for

$$x^3 \equiv 1 - y_0^2 \mod p \tag{1}$$

By HW8 Problem 6(e), since $p \equiv 2 \mod 3$, every integer has a *unique* cube root mod p. Therefore, there is a unique value x_0 which solve Congruence (**??**). In particular, not only is (x_0, y_0) a point on E, it is the unique point with y-coordinate y_0. □

(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).

Proof. We first observe that $p \equiv 2 \mod 3$. So that:

$$2p - 1 \equiv 2 * 2 - 1 \equiv 3 \equiv 0 \mod 3.$$

In particular, $2p - 1$ is divisible by 3, so raising to the power of $\frac{2p-1}{3}$ is a reasonable thing to do. Next we compute:

$$\left(z^{\frac{2p-1}{3}}\right)^3 = z^{2p-1} = z^{1+2(p-1)} = z \cdot (z^{p-1})^2 \equiv z \mod p,$$

where the last step uses Fermat's little theorem. In particular, $z^{\frac{2p-1}{3}}$ is a cube root of z. But by HW8 Problem 6(e), cube roots are unique! □

(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.)

Proof. Lets first write —`textToPoint`.

```
i. Input: A string s.
ii. Set $y_0$ to be textToInt(s).
iii. Compute $z \equiv 1 - m^2 \mod p$, and let $x_0 = z^{\frac{2p-1}{3}}$ be its cube root.
iv. Output: The point $(x_0, y_0) \in E(\mathbb{F}_p)$.
```

Next let's do `pointToText`.

```
i. Input: A point $(x_0, y_0) \in E(\mathbb{F}_p)$.
ii. Compute s = intToText($y_0$).
iii. Output: The string s
```

□

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.

Proof. Here are a couple of ways to get around Alice's basic security protocol. First one could present their decryption protocol with $[-R, c_1, c_2]$. Then it would compute $-T = (x_T, -y_T)$, and return $m_1 = x_T^{-1}c_1$ and $-m_2 = -x_T^{-1}c_2$.

Slightly less silly would be presenting the decryption protocol with $[R, kc_1, lc_2]$ for $k, l \in \mathbb{F}_p^*$. Then the decryption protocol would return km_1 and lm_2 from which m_1 and m_2 would easily be recovered. □

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 $\#E(\mathbb{F}_p)$).

Proof. To get around these security protocols and give Alice's decryption machine something it can't distinguish as related to Bob's message, we choose some k which is coprime to $\#E(\mathbb{F}_p)$, and compute $R' = kR$. We also choose random $c_1', c_2' \in \mathbb{F}_p^*$. We then present the decryption protocol with $[R', c_1', c_2']$. It computes $T' = n_A R'$, and presents us with $(m_1', m_2') = (x_{T'}^{-1}c_1', y_{T'}^{-1}c_2')$. Since we know c_1' and c_2' we can therefore recover $T' = (x_{T'}, y_{T'})$. But then:

$$T' = n_A R' = n_A kR = k(n_A R) = kT.$$

Therefore if we compute $k^{-1} \mod \#E(\mathbb{F}_p)$ we have (as in problem 5 and 7) $k^{-1}T' = k^{-1}kT = T$. Therefore we have obtained the encryption keys (x_T, y_T), and can invert them and decrypt $(m_1, m_2) = (x_T^{-1}c_1, y_T^{-1}c_2)$. $\square$