Part 4 wasn't too difficult following the pseudo-code specifications of the project details. The only problem was that I was attempting this problem in the dark because I had no idea if Part 3 public and private keys were working at the time. Initially I didn't know my Part 3 was the issue, therefore I messed around with Part4 pieces a lot.

The only problem I ran into with Part 4 was that it required a lot of KMACXOF256's to generate pseudorandom numbers to XOR with the message. I had a hard time keeping track of all the KMAC's that were happening and at a certain point I had issue with exponentiation because I was too focused on G which was Point(x,y) of the curve when X is equal to 18. I used that for other points on accident until I realized they were something completely different. For example the Public key would be a Point(x, y) of the V.x and V.y rather than the Point(18, y).

Once part 3 was solved, this section worked right after. Everything went smoothing following the pseudo-code in specifications. My only disclaimer is the GUI portion of part 4 requiring the user to type in the cryptogram of Z.x and Z.y which is displayed when a file is encrypted. The values are not saved anywhere and only displayed on the console.

There are files file.txt, file_enc.txt, file_dec.txt and pk.txt.

Pk.txt contains the public key x and y values generated from Part 3.

File.txt is an ordinary file with random sentences to be encrypted with the public key.

File_enc.txt is the encrypted file.

File_dec.txt is the decrypted file.

File.txt and File_dec.txt contents are exactly the same, therefore the decryption works.