Tasks

Elliptic Curve Cryptography (ECC) is a form of public-key cryptography that we have not discussed in detail. You have two main tasks.

1. Learn about the main operations involved in ECC (point addition, point doubling, and point multiplication). Understand how to implement these operations.

2. Complete the provided starter code in the language of your choice (C++, Java, or Python). The functions to complete are contained in the EllipticCurveArithmetic and the EllipticCurveCryptography classes. In EllipticCurveArithmetic, you must implement the modular inverse and ECC arithmetic operations (`add`, `addWithItself`, and `multiply`). In EllipticCurveCryptography, you must implement the key generation, encryption, and decryption functions. For these functions, we will be using the elliptic curve analog of the ElGamal system, with the hard-coded elliptic curve $y^2 = x^3 + 17x + 50$ over the finite field GF(191). You can find the scheme described in the 1987 paper *Elliptic Curve Cryptosystems* by Neal Koblitz [1]. You are welcome to write additional helper functions, but it is not necessary for completing the assignment. You should only modify the files associated with the EllipticCurveArithmetic and EllipticCurveCryptography classes (you can also modify the main file as necessary in order to encrypt and decrypt the provided plaintext and ciphertext). You are not permitted to use any libraries that have not been imported already. You are not permitted to collaborate with others in completing this assignment. Cite any sources you use in comments above the associated code.

Provided Files

The following files have been provided to you.

- (In C++/Java/Python) A main file and the files associated with the classes Point, Key, Helper, EllipticCurveArithmetic, and EllipticCurveCryptography.

- Two test files unique to you. Given your GT username, the test files will be named with the format `[username]_test1.txt` and `[username]_test2.txt`. An example of each is given below.

`test1` Example

```
41 179
72 131
ltdduvb5cv4a6na0fcbwg53kiz1xjukoq3snhj7job87aouqxmx8tcc7dku5m238
```

Line 1 contains the coordinates of the generator point G.
Line 2 contains the coordinates of the receiver's public key.
Line 3 contains a plaintext message to be encrypted.

`test2` Example

```
155 155
108
byaabddibhtgbkxmaydfaeihbprtcbhlammraaaaagpeadlsbudobrtuakssawrx
```

Line 1 contains the coordinates of the generator point G.
Line 2 contains the receiver's private key.
Line 3 contains a ciphertext message to be decrypted.

Submission

You must submit the following three files.

- A file named `report.txt`. This file should be formatted as follows.

  ```
  [Your GT username]
  [Encrypted plaintext]
  [Decrypted ciphertext]
  ```

  The encrypted plaintext should be the ciphertext produced when the `Encrypt` function is called on the plaintext and public key given in the `test1` file, and the generator point G is set to the point given in this same file. The decrypted ciphertext should be the plaintext produced when the `Decrypt` function is called on the ciphertext and private key given in the `test2` file, and the generator point G is set to the point given in this same file.

- The files associated with the classes EllipticCurveArithmetic and EllipticCurveCryptography. These will be named `EllipticCurveArithmetic.[*]` and `EllipticCurveCryptography.[*]`. Replace `[*]` with `h`, `java`, or `py`, depending on the language you choose.

References

[1] Neal Koblitz. "Elliptic curve cryptosystems". In: *Mathematics of Computation* 48.177 (Jan. 1987), pp. 203–203. DOI: 10.1090/s0025-5718-1987-0866109-5.