HW5_EE_4940_code

November 23, 2024

[1]:
```
#Import libraries
import sympy as sp
import numpy as np
from decimal import *
np.set_printoptions(legacy='1.25')
```

0.1 Problem 1

You can ignore the work in here, this is just for my own personal understanding.

[ ]:
```
a = 5
p = 19
n = 11
for x in range(0,p) :
    a_x = a**x % p
    a_n_x = a**(n*x) % p
    print(f'x = {x} | {a_x} | {a_n_x}')
```

```
x = 0 | 1 | 1
x = 1 | 5 | 6
x = 2 | 6 | 17
x = 3 | 11 | 7
x = 4 | 17 | 4
x = 5 | 9 | 5
x = 6 | 7 | 11
x = 7 | 16 | 9
x = 8 | 4 | 16
x = 9 | 1 | 1
x = 10 | 5 | 6
x = 11 | 6 | 17
x = 12 | 11 | 7
x = 13 | 17 | 4
x = 14 | 9 | 5
x = 15 | 7 | 11
x = 16 | 16 | 9
x = 17 | 4 | 16
x = 18 | 1 | 1
```

[ ]:
```python
p = 5
X = []
for a in range(1,p):
    i=1
    for x in range(1,p) :
        a_x = a**x % p
        #print(f'x = {x} | {a_x} | {a_n_x}')
        if a_x == 1 :
            X.append(i)
            break
        i+=1

print(X)
res = []
for val in X :
    if val not in res:
        res.append(val)

print(res)

print(f'{len(res)}')
```

```
[1, 4, 4, 2]
[1, 4, 2]
3
```

0.2 Problem 3

[ ]:
```python
def find_modulo_inverse(input, modulo) : #Finds inverse in modulo.
    #e.g. 2 inverse in mod 3 is 2 b/c 2*2 = 4(mod 3) = 1
    #==Inputs==
    #input: value to find the inverse of
    #modulo: value of the modulus

    i=modulo
    while i>0 :
        if np.mod(input*i,modulo) == 1 :
            return i
        i-=1
    return 0

def Elliptic_Add_GF(point_P, point_Q,a,b,p) : #

    if point_P == [0,0]: return point_Q #O + Q = Q
    if point_Q == [0,0]: return point_P #O + P = P

    if point_P == point_Q: # P + P = 2P
        inv = find_modulo_inverse((2*point_P[1]),p) #denominator of S
```

```
        S = (((3*point_P[0]*point_P[0]) + a) * inv)
        #print('same')
    else:
        inv = find_modulo_inverse((point_Q[0] - point_P[0]), p) #denominator of␣
↪S
        S = ((point_Q[1] - point_P[1]) * inv) % p

    if inv == 0 : #cannot divide by 0. --> occurs when P + Q = O
        return [0,0]

    x_R = (S*S - point_P[0] - point_Q[0]) % p
    y_R = (S*(point_P[0]-x_R)-point_P[1]) % p

    return [x_R,y_R]
```

[ ]:
```
nB = 8 #bob private key

PB = [16,5]
for i in range(1,nB) :
    PB = Elliptic_Add_GF(PB,G,a,b,p)
    #print(PB)

print('')
print(f'PB = {PB}')
```

```
PB = [12, 17]
```

[ ]:
```
xA = 4
M = [4,5]

C1 = [16,5]
for i in range(1,xA) :
    C1 = Elliptic_Add_GF(C1,G,a,b,p)

public_product = PB #xA*PB
for i in range(1,xA) :
    public_product = Elliptic_Add_GF(public_product,PB,a,b,p)
    print(public_product)

print(f'pub_product = {public_product}')

C2 = Elliptic_Add_GF(M,public_product,a,b,p)

C = [C1,C2]
print(C)
```

```
[17, 0]
```

```
[12, 6]
[0, 0]
pub_product = [0, 0]
[[19, 20], [4, 5]]
```

[ ]:
```
#Alice public key
C1 = C[0]
C2 = C[1]
print(C1)
nB_x_C1 = C1
for i in range(1,nB) :
    nB_x_C1 = Elliptic_Add_GF(nB_x_C1,C1,a,b,p)
    print(nB_x_C1)

nB_x_C1[1] =-nB_x_C1[1]
print(nB_x_C1)

M = Elliptic_Add_GF(C2, nB_x_C1,a,b,p)
print(M)
```

```
[19, 20]
[12, 17]
[18, 10]
[17, 0]
[18, 13]
[12, 6]
[19, 3]
[0, 0]
[0, 0]
[4, 5]
```

0.3 Problem 5

(a) Write a computer program to compute a power of a number in $\mathbb{Z}_p$. Your program should take a, n, and p and compute $a^n \pmod p$. Note that a and n may be very large so that an is infeasible to be evaluated. Recall the efficient approch based on the binary expansion of n, we discussed in class.

[2]:
```
def exp(a,n,p) : # computes a^n (mod p)
    output = 1
    while n > 0:
        if (n & 1) == 1 : output = (output * a) % p # if leftmost bit = 2
        a = (a * a) % p
        n >>= 1 # find next leftmost bit.
    return output
```

(b) Implement an encryptor for an RSA system with public key (n,e).

```
[ ]: def RSA_encrypt(M,e,n): #encrypts RSA message M by doing M^e (mod n)
         output = exp(M,e,n)
         return output
```

(c) Let M be your UMN student ID. Use your program to encrypt M with $n = 31189420800514467447616631563$ and $e = 28879207836360367989641236 03$.

```
[ ]: id = 5631519
     n = 31189420800514467447616631563
     e = 28879207836360367989641236 03

     C = RSA_encrypt(id,e,n)
     print(f'C = {C}')
```

```
C = 31027703950070711403380330146
```