Homework3

September 29, 2020

[11]:
```
########## Preamble:
########## We need fast powering, and extended euclidean algorithm in order to␣
↪find inverses.

def fastPowerSmall(g,A,N):
    a = g
    b = 1
    while A>0:
        if A % 2 == 1:
            b = b * a % N
        A = A//2
        a = a*a % N
    return b

def extendedEuclideanAlgorithm(a,b):
    u = 1
    g = a
    x = 0
    y = b
    while true:
        if y == 0:
            v = (g-a*u)/b
            return [g,u,v]
        t = g%y
        q = (g-t)/y
        s = u-q*x
        u = x
        g = y
        x = s
        y = t

def findInverse(a,p):
    inverse = extendedEuclideanAlgorithm(a,p)[1] % p
    return inverse
```

[7]:
```
########## Problem 1
##### Part (a)
```

```python
print('\n',"Part (a)",'\n')
x = "Hello World"
for i in x:
    print(i)

##### Part (b)
print('\n',"Part (b)",'\n')
x = "Hello"
y = x + '!'
print(y)

##### Part (c)
print('\n',"Part (c)",'\n')
print(ord('a'))
print(ord('A'))
print(ord(' '))
print(ord(','))
#print(ord(a))   Ord needs input of a char which needs ' '
#print(ord("hello")) Ord needs input of a char not a string.

##### Part (d)
print('\n',"Part (d)",'\n')
print(chr(98))
print(chr(40))
print(chr(ord('F')))
print(ord(chr(201)))
```

```
 Part (a)

H
e
l
l
o
 
W
o
r
l
d

 Part (b)

Hello!

 Part (c)
```

```
97
65
32
44

 Part (d)

b
(
F
201
```

[10]:
```
########## Problem 2
##### Part (a)
def textToInt(words):
    number = 0
    i = 0
    for letter in words:
        number += ord(letter)*(256**i)
        i+=1
    return number

##### Part (b)
def intToText(number):
    words = ""
    while number>0:
        nextLetter = number % 256
        words += chr(nextLetter)
        number = (number-nextLetter)/256
    return words

##### Part (c)
print(textToInt("Hello World!"))
print(intToText(10334410032597741434076685640))
print(intToText(157690076402712651527241))
```

```
10334410032597741434076685640
Hello World!
It worked!
```

[21]:
```
########## Problem 3
#We save a prime as a variable which we may vary:
p = 370141817103067776979133

##### Part (a)
def encrypt(m,k):
    return m*k % p

def decrypt(c,k):
    kinv = findInverse(k,p)
    return c*kinv % p

##### Part (b)
print('\n',"Part (b)",'\n')

key = 1479559274736299583l6
cipher = 8544984868677525224 5536

message = decrypt(cipher,key)
print(intToText(message))

##### Part (c)
print('\n',"Part (c)",'\n')


message = textToInt("Hello!")
print("Mesage as integer:",message)
myKey = 1213141516171819
cipher = encrypt(message,myKey)
print("Encrypted integer:",cipher)
newMessage = decrypt(cipher,myKey)
print("Decrypted integer:",newMessage)
print("Decoded to text:",intToText(newMessage))

##### Part (d)
print('\n',"Part (d)",'\n')
#First we update the prime:
p = 23169331
print("My new prime is",p)
#We'll then run the same code:

message = textToInt("Hello!")
print("Mesage as integer:",message)
myKey = 1213141516171819
cipher = encrypt(message,myKey)
print("Encrypted integer:",cipher)
newMessage = decrypt(cipher,myKey)
print("Decrypted integer:",newMessage)
print("Decoded to text:",intToText(newMessage))
print("We got the wrong answer because the message doesnt fit in F_p^*")
```

```
Part (b)
```

```
Testing…

 Part (c)

Mesage as integer: 36762444129608
Encrypted integer: 29808642902613194360915
Decrypted integer: 36762444129608
Decoded to text: Hello!

 Part (d)

My new prime is 23169331
Mesage as integer: 36762444129608
Encrypted integer: 16329898
Decrypted integer: 14171873
Decoded to text: á>Ø
We got the wrong answer because the message doesnt fit in F_p^*
```

[24]:
```python
########## Problem 4
#We can put this all in one function:
def doDiffieHellman(prime,generator,aliceSecret,bobSecret):
    #they compute public numbers to swap
    alicePublic = fastPowerSmall(generator,aliceSecret,prime)
    bobPublic = fastPowerSmall(generator,bobSecret,prime)
    print("Alice's public number is",alicePublic)
    print("Bob's public number is",bobPublic)

    #they both compute the shared secret
    aliceSharedSecret = fastPowerSmall(bobPublic,aliceSecret,prime)
    bobSharedSecret = fastPowerSmall(alicePublic,bobSecret,prime)
    print("Alice computed",aliceSharedSecret)
    print("Bob computed",bobSharedSecret)


##### Part (a)
print('\n',"Part (a)",'\n')

#We first choose a prime and generator
p = 17
g = 3

#Alice and bob have secret numbers
a = 5
b = 3

doDiffieHellman(p,g,a,b)
```

```
##### Part (b)
print('\n',"Part (b)",'\n')

#We first choose a prime and generator
p = 56509
g = 2

#Alice and bob have secret numbers
a = 3482
b = 20487

doDiffieHellman(p,g,a,b)

##### Part (c)
print('\n',"Part (c)",'\n')

#We first choose a prime and generator
p = 370141817103067776979133
g = 31415926535

#Alice and bob have secret numbers
a = 112233445566778899
b = 998877665544332211

doDiffieHellman(p,g,a,b)
```

```
 Part (a)

Alice's public number is 5
Bob's public number is 10
Alice computed 6
Bob computed 6

 Part (b)

Alice's public number is 49892
Bob's public number is 18602
Alice computed 31046
Bob computed 31046

 Part (c)

Alice's public number is 234986677042258789314441
Bob's public number is 357420017376256037149781
Alice computed 221206967657463523352818
```

```
Bob computed 221206967657463523352818
```

[0]:

```
```