Takehome 1

October 24, 2021

[2]:
```
########## Preamble
########## Loading in fastpowering and euclidean algorithm and find inverse


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

def textToInt(words):
    number = 0
    i = 0
    for letter in words:
        number += ord(letter)*(256**i)
        i+=1
    return number

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

[3]:
```
########## Problem 2
def findRoot(c,e,p,q):
    d = findInverse(e,(p-1)*(q-1))
    m = fastPower(c,d,p*q)
    return m
```

[4]:
```
########## Problem 3

#####Part (a)
def millerRabin(a,n):

    #first throw out the obvious cases
    if n%2 == 0 or extendedEuclideanAlgorithm(a,n)[0]!=1:
        return True

    #Next factor n-1 as 2^k m
    m = n-1
    k = 0
    while m%2 == 0 and m != 0:
        m = m//2
        k = k+1
    #Now do the test:
    a = fastPowerSmall(a,m,n)
    if a == 1:
        return False

    for i in range(0,k):
        if (a + 1) % n == 0:
            return False
        a = (a*a) % n

    #If we got this far a is not a witness
    return True
```

```
#####Part (b)
# This function runs the Miller-Rubin test on 20 random numbers between 2 and
↪p-1.  If p is composite there is only a (1/4)^20 probability it will return
↪true.
def probablyPrime(p):
    for i in range(0,20):
        a = ZZ.random_element(2,p-1)
        if millerRabin(a,p):
            return False
    return True

#####Part (c)
def findPrime(lowerBound,upperBound):
    while True:
        candidate = ZZ.random_element(lowerBound,upperBound)
        if probablyPrime(candidate):
            return candidate

#####Part (d)
p1 = findPrime(10,100)
p2 = findPrime(1000,10000)
p3 = findPrime(10**99,10**100)
p4 = findPrime(10**499,10**500)

print("Is",p1,"prime?",p1 in Primes())
print("Is",p2,"prime?",p2 in Primes())
print("Is",p3,"prime?",p3 in Primes())
print("Is",p4,"prime?",p4 in Primes())
```

```
Is 29 prime? True
Is 4241 prime? True
Is 11294667217381905761910005777991975563515594336089003702450374898297270940892
40826828046738035278301 prime? True
Is 96104496511259421695654892201987309450138921010835641096632548270879418936976
84816263703515403212260773714151512843465939684325099406621342527547966377779207
50715385803425215174866396558590558579928998812773297074999048628649149059288539
20947288526673938641621987629567723963195103077539609057275433475027278353579706
64262468263590432084235782400279848632934823928098169132516253534015005113907586
15425046986023435257187797846656891550222641629859125559513908713400365156822475
73797141304766529038027 prime? True
```

```
[6]: ##########Problem 4
#####Part (a)
def generateRSAKey(b):

    #Generate some primes
    p = findPrime(2^(b-1),2^b)
    q = findPrime(2^(b-1),2^b)
    N = p*q
    M = (p-1)*(q-1)
    #next lets find an encryption exponenet
    while True:
        e = ZZ.random_element(2,M-1)
        gcd = extendedEuclideanAlgorithm(e,M)
        if gcd[0]==1:
            d = gcd[1] % M
            break
    publicKey = [N,e]
    privateKey = [N,d]
    return[publicKey,privateKey]

#####Part(b)
def RSAEncrypt(message,PublicKey):
    return fastPowerSmall(message,PublicKey[1],PublicKey[0])

def RSADecrypt(cipher,PrivateKey):
    return fastPowerSmall(cipher,PrivateKey[1],PrivateKey[0])
```

[10]:
```
#########Question 5

### Part (a)
keys = generateRSAKey(16)
print("my keys are",keys)
smallPublicKey = keys[0]
smallPrivateKey = keys[1]

m = 314159

c = RSAEncrypt(m,smallPublicKey)
print("Ciphertext is:",c)

m1 = RSADecrypt(c,smallPrivateKey)
print("Decyphered message is:",m1)


### Part (b)
pubKey,privKey = generateRSAKey(512)
print("My public key which I will post to discord is:",pubKey)
print("My private key which I won't is: ",privKey)
print("Don't delete it!")
```

```
my keys are [[2787545483, 2019911167], [2787545483, 68449543]]
Ciphertext is: 2656250519
```

```
Decyphered message is: 314159
My public key which I will post to discord is: [93160489447278628018556404190327
66024815618077673779511628481072492260926402309365688244168718270327403964370293
62668486175965556540155377998571993081060353870556399870682555640999805682532976
48561016020439624202128361541882271727061631072906425043040050610939530219303856
694769187673770180563699916769257643, 227069248966942210225446217171091807478325
06518882522137208469847278158664341519454460296544274047442241105238616669026608
05422591918584807027395840049544167429617843291510345376361691852542241476652350
43939583454979531394629678400326576414974510358490525014315320316541714031724196
82550200501058427794932695]
My private key which I won't is:  [931604894472786280185564041903276602481561807
76737795116284810724922609264023093656882441687182703274039643702936266848617596
55565401553779985719930810603538705563998706825556409998056825329764856101602043
96242021283615418822717270616310729064250430400506109395302193038566947691876737
70180563699916769257643, 6855650022263014314963720270614697812409222551007907509
72617692896782050132001376396497398862240289553615528782783909728353472594966422
01048989809743475663771246282184520819853789715494836562246181872783324578581280
56824390130809505366362368458334066739836543014666167223685721991934458871155472
1444121340359]
Don't delete it!
```

[12]:
```
###Part (c) COMMUNICATION

myPub =␣
 ↪[93160489447278628018556404190327660248156180776737795116284810724922609264023093656882444168
 ↪2270692489669422102254462171710918074783250651888252213720846984727815866434151945446029654
myPriv =␣
 ↪[93160489447278628018556404190327660248156180776737795116284810724922609264023093656882444168
 ↪6855650022263014314963720270614697812409222551007907509726176928967820501320013763964973988

#I got sent:
cipher =␣
 ↪5427014709442861408939420843460244780511921581448565381930985171919133876276176899760445176

print(intToText(RSADecrypt(cipher,myPriv)))
```

```
MESSAGE FOR: GabrielDH. Your secret number is 1. Respond with your full name and
triple your secret number.
```

[14]:
```
#####Continued:

theirPub =␣
 ↪[126654416197366733724065827912654338262932627125666418602498017264808263590587336703888247
 ↪1134874730012108494103649774031255544406990243163542342729105750691125335237155837982452977

messageText = "My name is Gabriel Dorfsman-Hopkins.  Triple my number is 3!"
message = textToInt(messageText)
```

```python
cipher = RSAEncrypt(message,theirPub)
print("Post this cipher to Discord:",cipher)
```

```
Post this cipher to Discord: 715272601850757632593158611263376129108945717956110
83080808874721780392807181518392769887752444110511924517255581778244164417744020
86705390343780552015864275458734438868509482697118410862289718740786059957865676
28528568413301541922934829199953324583215715083893427661444051562509246510543828
32878815627448271
```

```
[0]:
```