Python project proposal

Functions needed for this program:

ifprime(n): This function is able to determine if a number is a prime by testing all integers from 2 to n-1. If none of these is a divisor of n, then n is a prime number.

assign result = True

for x in range(2,n): # check every integer between 1 and n

if n is divisible by x:

result will = False

this loop will break

return result

quotient(a,d): takes in any integer a and a positive integer d, and return a//d and a%d.

return (a//d, a%d)

euclidean(a,b): This is a recursive function which can apply the Euclidean algorithm to a and b. It can return (d, s, t) which d is the gcd(a,b), s, t are solution to the linear equation: as + bt = gcd(a,b).

q, r = quotient(a,b)

if r == 0, this means now the divisor is the gcd(a,b):

return ( the divisor b, r, 1)

else:

d1, s1, t1 = Euclidean(b,r) # replace a with b and replace b with r

d = d1

s = t1

t = s1-t1\*q # the proof of this part is done in the second python assignment

return (d, s ,t)

find\_inverse(s,t,a,b): For the sake of solving the root of a congruence equation, we have to make sure that s, t in the linear equation: as + bt = gcd(a,b) satisfy the condition which s <0 and t>0. This function will return t, the inverse of a modulo b, when this condition is met.

While s > 0 or t < 0 (the negation of the condition), then WLOG we use the linear equation theorem, ( s + k\*(b/(gcd(a,b)), t – k\*(a/(gcd(a,b)), to meet the condition.

s -= b # k < 0

t += a

we stop until the condition is met.

Return t (the inverse)

decrypt(n,p,q,k): decrypt the message, n, using the public key and the private key.

Q = (p-1)\*(q-1)

x, s, t = euclidean(Q,k)

t = find\_inverse(s, t, Q, k) # we find the inverse in these 2 steps using defined function: Euclidean and find\_inverse.

x is congruent to n\*\*(t) (mod p\*q)

reduce n\*\*(t) to an integer y

return y

Menu:

To make a letter-to number table, import string.

Assign “letters” = string.ascii\_uppercase

Use a for loop to run to traverse through “letters” to create 2 string. One to be the first column that contains all the upper-case letters. The other contains the number that’s matched to those letters.

Print(string1)

Print(string2)

What just printed is the table that our user will see

Create 2 dictionaries as these dictionaries are for the programmer to see and use. One is letter-to-number and the other is number-to-letter.

Ask if the user want to be Alice or Bob, and what’s the difference between them.

While user does not enter “quit”:

If user == Alice:

Ask the user to pick a good private key and a good public key.

Use ifprime function to determine if these are all prime

If not, tell the user which one is not prime.

Ask the user what’s the encrypted message.

Decrypt the message.

While decrypting, only decrypt the numbers, and leave the spaces and punctuation marks aside.

Use the number-to-letter dictionary to translate what’s decrypted into English.

Print the decrypted message for the user.

elif user == Bob:

Ask the user what the public key is.

Ask the user what message that he/she wants to encrypt.

Use the letter-to-number dictionary to translate the letters into numbers

Encrypt the message.

While encrypting, only encrypt the letters, and keep all the punctuation marks and spaces.

Encrypt len(str(m))-1 digit at a time, ex. If m = p\*q = 1067, then we maximum encrypt 3-digit number at a time.

Encrypting is easy, so we do not need another function to do it. It is just x\*\*(k) = code (mod m) which the encrypted code is the “code” here after reducing.

Print the encrypted message.

Print ( “Bye bye”) to the user if he/she types “quit”:.