**18CA314-Cryptography and Network Security**

**ASSIGNMENT-1**

**Name:Abhishek.s**

**Rollno:AM.SC.p2csc19003**

1. **Find the multiplicative inverse of all the elements in Z5 and Z11**

**Ans:** Multiplicative Inverse of:

Z5->

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | 1 | 2 | 3 | 4 |
| a-1 | 1 | 3 | 2 | 4 |

Z11->

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| a-1 | 1 | 6 | 4 | 3 | 9 | 2 | 8 | 7 | 5 | 10 |

1. **Determine the gcd of 56245 and 43159**

**Ans:** 56245=43159\*1+13086

43159=13086\*3+3901

13086=3901\*3+1383

3901=1383\*2+1135

1383=1135\*1+248

1135=248\*4+143

248=143\*1+105

143=105\*1+38

105=38\*2+29

38=29\*1+9

29=9\*3+2

9=2\*4+1

2=**1**\*2+0

Therefore, gcd(56245,43159)=1.

1. **Compute phi(n) for 34 and 210**

**Ans:** According to Euler’s product formula

Phi(34)=34\*(1-(1/3))

=81\*2/3 =54.

Phi(210)=210\*(1-(1/2))

=1024\*1/2 =512.

1. **Compute 3100 mod(31319) Ans:** Here e=100 =>26+25+22

30 mod 31319=3

32 mod 31319=9

34 mod 31319=81

38 mod 31319=6561

316 mod 31319=14418

332 mod 31319=21979

364 mod 31319=12185

3100 mod(31319)=12185\*21979\*81 mod 31319

=5346\*81 mod 31319

=25879.

**Part B -Programming Assignment**

1. Write a program to implement Extended Euclidean Algorithm and find multiplicative inverse for following values.

(a) 53947−1mod 56211

(b) 19385−1mod 43159

def ext\_euc(a, b):

s = 0

old\_s = 1

t = 1

old\_t = 0

r = a

old\_r = b

while b != 0:

q = a // b

(a, b) = (b, a % b)

(s, old\_s) = ((old\_s - (q \* s)), s)

(t, old\_t) = ((old\_s - (q \* t)), t)

if old\_s < 0:

old\_s += old\_r

if old\_t < 0:

old\_t += r

return old\_s

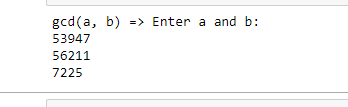
print("gcd(a, b) => Enter a and b: ")

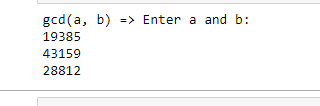
a = int(input())

b = int(input())

print(ext\_euc(a, b))

**output:**





2. In cryptography, a brute-force attack consists of an attacker submitting many passwords or passphrases with the hope of eventually guessing correctly. Implement a DES algorithm in any programming language(You are free to use language libraries). and decrypt the following cipher text using brute-force attack. Convert the hexadecimal value to string in the final stage. Cipher text: 0x4B518774A408E3E5

3.In real world, the commonly used RSA key size if 1024 bits, which is hard for cryptanalysis with limited resources. Implement a RSA algorithm with integer data type and show that you are able to decrypt the cipher text without knowing the private key

import random

def ext\_euc(a, b):

s = 0

old\_s = 1

t = 1

old\_t = 0

r = a

old\_r = b

while b != 0:

q = a // b

(a, b) = (b, a % b)

(s, old\_s) = ((old\_s - (q \* s)), s)

(t, old\_t) = ((old\_s - (q \* t)), t)

if old\_s < 0:

old\_s += old\_r

if old\_t < 0:

old\_t += r

return old\_s

def check\_if\_prime(val):

if val == 2:

return True

if val < 2 or val % 2 == 0:

return False

for i in range(3, val):

if val % i == 0:

return False

return True

def gcd(a, b):

while b != 0:

a, b = b, a % b

return a

def main():

print("Enter p & q :")

p = int(input())

q = int(input())

msg = int(input("Enter value to be encrypted: "))

if check\_if\_prime(p) and check\_if\_prime(q):

n = p \* q

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

e = random.randrange(1, phi)

g = gcd(e, phi)

while g != 1:

e = random.randrange(1, phi)

g = gcd(e, phi)

d = ext\_euc(e, phi)

print("private key: " + str(d) + " & public key " + str(e))

c = (pow(msg, e) % n)

print("Encrypted value is : " + str(c))

m = pow(c, d) % n

print("Decrypted value is : " + str(m))

if \_\_name\_\_ == '\_\_main\_\_':

main()

**output:**

