Lab – 2

Subject : NIS

Aim :- Aim is to develop the code of Multiplicative inverse, Multiplicative Cipher and to combine previous both the code implement Affine Cipher.

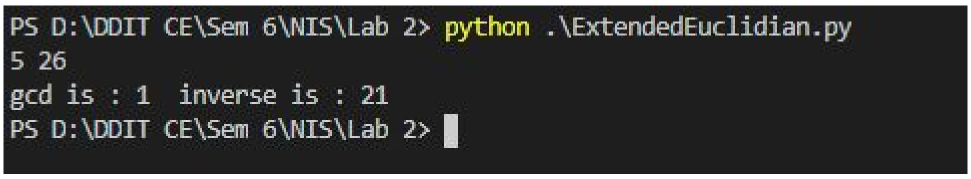
1.Implement the code to find multiplicative modular inverse

There are two integers ex. A and M if we want to find the modular multiplicative inverse of A with modulo M. assume the inverse is k. -Formula is a * k ~= 1 (mod M). The multiplicative inverse of A modulo M is exists if and only if a and m are co-prime i.e The gcd of A and M is 1. If the gcd is not equivalent to 1 then there is no inverse exists.

Program: -

```
def multiplicative_inverse(a,n):

    r,t,t1,t2=0,0,0,1

    r1=n

    r2=a

    while(r2 > 0):

        q=r1//r2

        r=r1-q*r2

        r1=r2

        r2=r

        t=t1-q*t2

        t1=t2
        t2=t

    return r1,t1


a,n=map(int,input().split())

gcd,inverse=multiplicative_inverse(a,n)

inverse=inverse%n

if( gcd != 1):

    inverse=-1 #Here,-1 shows that Inverse is not possible

print("gcd is : "+str(gcd)+"  inverse is : "+str(inverse))
```

Output: -



```
PS D:\DDIT CE\Sem 6\NIS\Lab 2> python .\ExtendedEuclidian.py
5 26
gcd is : 1  inverse is : 21
PS D:\DDIT CE\Sem 6\NIS\Lab 2> 
```

2.Implement the code for encryption and decryption of Multiplicative cipher

This cipher is not strong enough, because in this cipher the formula is Encryption formula : (P * key) mod 26. Decryption formula : (C * key^-1) mod 26. In the range of 1 to 26 there are only 12 number possible as a key, that's why it is easy for attacker to get into it and intercept between sender and receiver. -Here key must have multiplicative inverse under the modulo

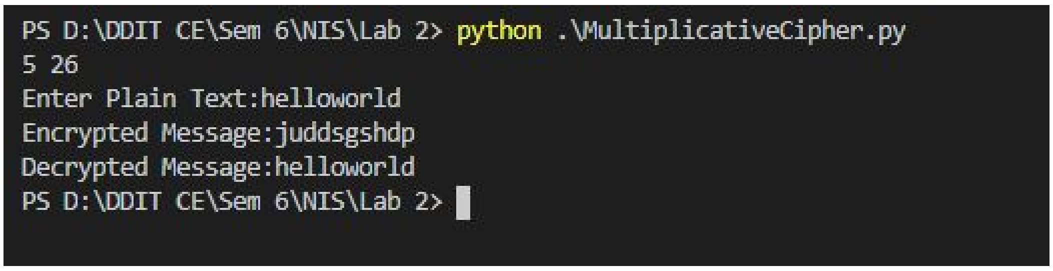
Program: -

```
def multiplicative_inverse(a,n):

    r,t,t1,t2=0,0,0,1

    r1=n

    r2=a
```

```python
    while(r2 > 0):

        q=r1//r2

        r=r1-q*r2

        r1=r2

        r2=r

        t=t1-q*t2

        t1=t2

        t2=t

    return r1,t1

def encrypt(msg,key):

    l=list(msg)

    l1=len(l)

    for i in range(l1):

        l[i]=chr(((ord(l[i])-97)*key)%26+97)

    return ("".join(l))

def decrypt(msg,key,n):

    l=list(msg)

    l1=len(l)

    gcd,inv = multiplicative_inverse(key,n)

    for i in range(l1):

        l[i]=chr(((ord(l[i])-97)*inv)%26+97)

    return ("".join(l))


key,n=input().split()

key=int(key)
```

```
n=int(n)

msg=input("Enter Plain Text:")

enc_msg=encrypt(msg,key)

print("Encrypted Message:"+enc_msg)

dec_msg=decrypt(enc_msg,key,n)

print("Decrypted Message:"+dec_msg)
```

Output: -



```
PS D:\DDIT CE\Sem 6\NIS\Lab 2> python .\MultiplicativeCipher.py
5 26
Enter Plain Text:helloworld
Encrypted Message:juddsgshdp
Decrypted Message:helloworld
PS D:\DDIT CE\Sem 6\NIS\Lab 2>
```

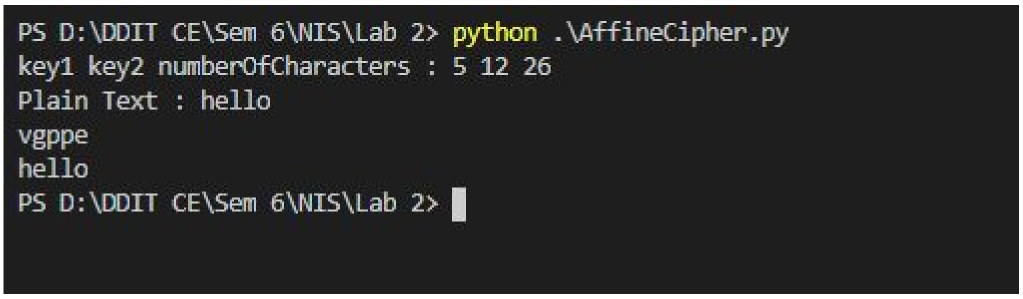
3.Implement the code of Affine Cipher, It is a combination of additive cipher and multiplicative cipher

Affine cipher is type of monoalphabetic substitution cipher, also it is a combination of multiplicative cipher and additive cipher. It uses modular arithmetic to transform the integer that each plaintext letter corresponds to into another integer that correspond to a ciphertext letter. The function for Encryption is ((P * key1) + key2) mod 26 .In this formula key1 must be has multiplicative inverse under the mod. And key2 belongs to any 26 alphabet.There are 12 * 26 combination possible for guess pair of key.In deciphering the ciphertext, we must perform the inverse functions on the ciphertext to retrieve the plaintext. The first step is to convert each of the ciphertext letters into their integer values. The function for Decryption is ((C – key2) * key1 ^ -1) mod 26 .This is also not strong enough.

Program: -

```
def multiplicative_inverse(a,n):

    r,t,t1,t2=0,0,0,1

    r1=n

    r2=a

    while(r2 > 0):

        q=r1//r2

        r=r1-q*r2

        r1=r2

        r2=r

        t=t1-q*t2

        t1=t2

        t2=t

    return r1,t1

def encrypt(msg,k1,k2):

    l=list(msg)

    l1=len(l)

    for i in range(l1):

        l[i]=chr(((ord(l[i])-97)*k1+k2)%26+97)

    return ("".join(l))

def decrypt(msg,k1,k2):

    l=list(msg)

    l1=len(l)

    gcd,inv = multiplicative_inverse(k1,n)

    for i in range(l1):

        l[i]=chr(((ord(l[i])-97-k2)*inv)%26+97)
    return ("".join(l))


k1, k2, n = list(map(int, input("key1 key2 numberOfCharacters : ").split()))

msg = input("Plain Text : ")

enc_msg=encrypt(msg,k1,k2)

print(enc_msg)

dec_msg=decrypt(enc_msg,k1,k2)

print(dec_msg)
```

Output: -



```
PS D:\DDIT CE\Sem 6\NIS\Lab 2> python .\AffineCipher.py
key1 key2 numberOfCharacters : 5 12 26
Plain Text : hello
vgppe
hello
PS D:\DDIT CE\Sem 6\NIS\Lab 2> 
```