ReadMe File of Projects

- **Project 1 – Image Encryption & Decryption using AES**

  Python file has been attached.

  Importing libraries –

In [14]:
```
import base64       # module provides functions for encoding binary data to printable ASCII characters and decoding such enc
import hashlib      # module interface for hashing messages easily
import numpy as np  # provide an array object
import cv2          # opencv python library for image display (computer vision problems)
```

In [15]:
```
from Crypto.Cipher import AES
from Crypto.Random import new as Random   # to generate long integer with k random bits
from hashlib import sha256                # hash function with block size of 32 bits
from base64 import b64encode,b64decode    # encodes or decodes byte-like objects
```

Using AES –

In [16]:
```
class AESCipher:
    def __init__(self,data,key):
        self.block_size = 16          # 16 bytes block size
        self.data = data

        # encoding and then sending to sha256...return a digest object (to ensure right file is being evaluated)
        self.key = sha256(key.encode()).digest()[:32]

        # Padding some data so that it becomes the multiple of block size
        self.pad = lambda s: s + (self.block_size - len(s) % self.block_size) * chr (self.block_size - len(s) % self.block_s

        self.unpad = lambda s: s[:-ord(s[len(s) - 1:])]

# Encryption function -
    def encrypt(self):
        plain_text = self.pad(self.data)
        iv = Random().read(AES.block_size)
        cipher = AES.new(self.key,AES.MODE_OFB,iv)
        return b64encode(iv + cipher.encrypt(plain_text.encode())).decode()

# Decryption function -
    def decrypt(self):
        cipher_text = b64decode(self.data.encode())
        iv = cipher_text[:self.block_size]
        cipher = AES.new(self.key,AES.MODE_OFB,iv)
        return self.unpad(cipher.decrypt(cipher_text[self.block_size:])).decode()
```

In [17]:
```
with open("Crypto.jpg", 'rb') as img_file:
    BI = base64.b64encode(img_file.read())
```

In [21]:
```
path = r'C:\Users\sk19s\OneDrive\Desktop\Crypto.jpg'

# reading image in default mode
image = cv2.imread(path)

# window name
window_name = 'image'

# using cv2.imshow() method to display image
cv2.imshow(window_name, image)

# necessary to avoind python kernel from crashing
cv2.waitKey(0)

# closing all open windows
cv2.destroyAllWindows()
```

Output of displaying image (to be encrypted) using cv2 module –

Encryption –

```
In [19]:
1 enc = AESCipher(BI.decode('utf-8'),'sakshi').encrypt()
2 print(enc)
```

```
cO4/eurwtYSwXV3civbetI+tPjL392lcI8iiFX7kq8lw4oPpl2yjna22wdUFJ5PdPgBsam5jFn4GIFXsMFpbTxqaufRAaedYCUnq2t+QWD34+TPhts4JY77tFLtn5
Y5JaKA24ZnZZ4oTcxjeMQ2d4qggjnPNYP7MoX+Dujy1H49tPqz2/BBqgC94T9L2HVRMcMirqqUrQ0mkpkk3iTD10obwyQogwvQqOL5+xv46+Vw+fIAhwvrIGdy800
s7hnIMvQ1ZLO6a81nHOnD2PUq5P56uFDpMsF9csWysR8E46OowEP4LWIXOjs+bCCGCibuGgXW32ex9Xbe/5ZdafgMo+XzZBzIU1bJuD9As+oG1UnJOpVV/Xa9330b
iLm0LW4CvUh35bikaQA2WapfTJw8ICQCoAvu43MWi0lc+ZWW/WKg3mwxA5YGbUd4Q01zz7aLNI4IjazhID6Zv43ahN6jcy+jxH07ZuoKPcPB3JX+pcN164Z/jHb9L
tWUI/tVpSeynQwYUOQNkPZXtvea7vRkTvyrGGpvUMkvVzfDtGg6mIpFD3s90yMAdAzBYXzq2zEjpyj1Hhd/+PxYcGDKr46bANbiJVqAHtAofSzfB6dRSnm0UJ5PkN
J79s+keRiuffsfyxcx8wUEo0dFPjEZF3DTgwk+p26YlDwJ6SGBuB9WaoQSIKzApGr0r5bv4rcAH1r25dAOW8rnOo16uoagIKwIheFztLijac11kjCwUuADMtabRF+
kb2WD3d5EZPK7B1spWRk216//BlHKe/YfXJbsq6LVmbHRKoDzyGWEah1ccuUSLFPdE8HbWnYwyZHF+aWFXMkTE6kF2j5MsLKzcaPDme8ZlFLKpgrt/KqRvzgOis/I
xrW0XgsOLYcZvHViGTCgr8WgcLwe8XjqpdQLX1avgbNRcn8dbU+Un89bAQnOTcWjv14DB+RCSf4afZCrpAfyg91dtSvU73Hto/B2MTZrBOhrt0SSGsFU0C5l6KYn+
pL6WNWbRTHGxy++7DtXlrMSFISDDZB+oG8PTXF9M7Hs2RgK33hZn9IJ6d0ClPvebbd+x77u/IpEMSAg/GO+QMJxHJmb1PWxZo/b65ihftU7EcL66cyMgR+fTBECaA
HeSj7C3dbygJPLtDijMbazoUJjq88EviDUDGr4pgQaH6A1jGfcEZW7eD8uXqTOeAy3wzjiiLWe0dSGFts79leFYju+w3qoVyNuhzw8ncewtJ7CYZ7VYCJRkIxGfsC
fh+SxrcCb2W2FERG9/u6pobowe4N/7tXgE1rOpcGlOgvhWdwi5lmh1xB41hGFI74+gY1evG2KiWkWcWe5lzvcnuw2ts0xjFoS0cZJFsOrNaMSlAL6ZnkMc4dM3bFP
yf7rj2QIpCL7rY5lp51LIzQEg70eG90dMaq146Qo51gJnJ/BgeIIGJu+WoExwpUWk6aC/zHZX0ufn9sWUeJB/3WPLftG5fKqcYaoNO38rRQqb/pnXS/Jf7RCaPJjO
vU1yU2F2B74XBc0iRnHUOdDVS4kNuxw/2CozBXY0OlvklvsxcXsyRIfqP7AN5Jqg0YidtyFsPJRLuxZIgcaTcKJHpEovPjjG+UhEal1m1VPyP4WsvRz30kNOogi9z
tXSlYkaBVYi2VmxJTr9UWoWjfukyQI8R+eju1dmN6GXTzSzt2ha0MtC1Lk0XYgKRvN6LrG6c0LRdr/pPOHy/2GAP20NIPi54tmF9AJWBhjot8RClFDk26nO7ZYnh+
tGaj8tje4QzubeVWe3ufwEYpaCz222uulPaJSGGhMBmyOfKb+WO1wTrDPczyllFDdKr6Jrtu2Mxg6uaRGM7Jzx0dHUOawQ94G30SFOV1cbFBmD29f1pv9aOATl7/r
2XRdNGhNkCwQi/NFaw8b2TrNYOOfl9X1ja2/LdBrDqn5ONkavf8Iau6KIXEks7ZDIMpvTDCrvW8LW4pQmNelVSZcUFt/ZhVmyOeziFk375Yw8y3Py6P42fXhcSfHj
o+XiX+mAhlLsEdnl0a9BXgByaA/HNiXevNtZuXO6SOGTg1GbyU/3YNZ7HKO9WAfliZn+m/OTziWG+8UVV33jft2wH+8hxFKIIkYO3Ttuyn0AwcuMICdNr7yOZ5IIk
x8m9fcubmG+fuOQQY6/ElbNji/TGq5pG7ITn+nk3NcedhxrbsmcF6w5C34PBcz9tNtRXPqytjy6jfK3ctsFfkBolsHUqpRBkswVMvY5PIF4eCtLHSKXPka4ju0JrS
CHSWVDuC3wWutCdkWeDLWI15qO1QYSACdfH1qR8k6xZSXdKi2TLvrpcchJg7EIQ+XdHdiU0Zwhd5JeRiJIpbP4bUPLSKSiIM8IrJYEDmkuPOsjmZ3ZgnpCx5IJjyo
```

Decryption –

```
In [20]:
1 dec = AESCipher(enc,'sakshi').decrypt()
2 print(dec)
```

```
/9j/4RfmRXhpZgAATU0AKgAAAAgADAEAAAMAAAABB4AAAAEBAAMAAAABA+gAAAECAAMAAAADAAAAngEGAAMAAAABAAIAAAESAAMAAAABAAEAAAEVAAMAAAABAAMAA
AEaAAUAAAABAAAApAEbAAUAAAABAAAArAEoAAMAAAABAAIAAAExAAIAAAAiAAAAtAEyAAIAAAAUAAAA1odpAAQAAAABAAAA7AAAASQACAAIAAgALcbAAAAnEAAtxs
AAACcQQWRvYmUgUGhvdG9zaG9wIENDIDIwMTcgKFdpbmRvd3MpADIwMTg6MDc6MzAgMTU6MDY6MjQAAAAABJAAAAcAAAAEMDIyMaABAAMAAAAB//0AAKACAAQAAAA
BAAAHgKADAAQAAAABAAAD6AAAAAAAAAAGAQMAAwAAAAEABgAAARoABQAAAAEAAAFyARsABQAAAAEAAAF6ASgAAwAAAAEAAgAAAgEABAAAAAEAAAGCAgIABAAAAAEA
ABZcAAAAAAAAAEgAAAABAAAASAAAAAH/2P/tAAxBZG9iZV9DTQAC/+4ADkFkb2JlAGSAAAAAAf/bAIQADAgICAkIDAkJDBELCgsRFQ8MDA8VGBMTFRMTGBEMDAwMD
AwRDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAENCwsNDg0QDg4QFA4ODhQUDg4ODhQRDAwMDAwREQwMDAwMDBEMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDA
wM/8AAEQgAUwCfAwEiAAIRAQMRAf/dAAQACv/EAT8AAAEFAQEBAQEBAAAAAAAAAAMAAQIEBQYHCAkKCwEAAQUBAQEBAQEAAAAAAAAAAQACAwQFBgcICQoLEAABBAE
DAgQCBQcGCAUDDDMBAAIRAwQhEjEFQVFhEyJxgTIGFJGhsUIjJBVSwWIzNHKC0UMHJZJT8OHxY3M1FqKygyZEk1RkRcKjdDYX0lXiZfKzhMPTdePzRieUpIW0lcTU
5PSltcXV5fVWZnaGlqa2xtbm9jdHV2d3h5ent8fX5/cRAAICAQIEBAMEBQYHBwYFNQEAAhEDITESBEFRYXEiEwUygZEUobFCI8FS0fAzJGLhcoKSQ1MVY3M08SUGF
qKygwcmNcLSRJNUoxdkRVU2dGXi8rOEw9N14/NGlKSFtJXE1OT0pbXF1eX1VmZ2hpamtsbW5vYnN0dXZ3eHl6e3x//aAAwDAQACEQMRAD8A9VWRkfWbBqsLKmvvjl
7I26fuucRu/wCoRfrDa+vpduyR6haxxH7riA7/ADm+xYvTiMfByMm0UsD3Nqxn5LTsLnS279Kxvqem6r2+39GqHN81kjmjhxkQ9JyTySHFwx/lFu8ty8JYzlmDL1c
EYA8Nn+Unf6f1bD6hIpJbY0Sa3iHR+8I3Nc3+orq5PJH2X6wA07g71mFwPjaR6rWH86t3qLrFLyeeWUZIzrjxSMJGPyy/rMXM4Y4zCUL4MkeMA7x8Grl9RxsU7Xku
sOuxupjxd+a1Dx+sYl7ww7q3O0bviCfDc0uWXhsfmZ7XPgy/1LJ8Afo/yvzWKPUTU/Mt9PVn0SNu0At9rmj97+urTXekVXM6hjYbZtdLuzG6u/8AMU2I+67pzHB0X
OrIa92vuHta93737y4DLxusZXUzgW3102vt9J9hc5/vdw7hv0k2RkPliT5LonENcuSOMXQ4zw8Z7R/ednq/1sYGa2iqtxgMadT5Od9Jyy8X6zYb6wTZteRqDOit3/
Urp/Tse0lzsnNppdkMutg1ubWf1ljMf3Npc31K9n85/wCfESmnGt6YMbqOJTk3BxNb7WBzmVnVte5u1+7eX/n+xRmE64pEeTJHmsRzexjxzkADKWWuGEfD/C/rMMb
r9NwcK73O2mIBOpiVbx88WPJoc72EAmSIcROuq5XMHTsLLswcRk2Mr9YDH1e0zsePT/wnLH7UToeT1h/Um12Yl7sa4Fj73UWVhhAPpW2b2Nbt3e36SjBPVtGAqx+L
3NXV7m+15JPiY/uKtV9VbH6QT5j/AFKwK6nBrftLyHCZrBbuifbud7ms9qMbqq2+0trb4uMkp4nLuwyxx7fY7v7VxjAEz4EQoP6zjtMBpPmYH/klzb+pUSYdMTx5K
lf1hrIBcC530R5JHJLugYY9i9Nk/WMMH6KokcE8ws931izLHbfoeIAWNTl2WuIP0RzH3p3Oc/SIjT5JpySPVeMUB0dS3qme/QWkGSInt8lW+15bng+o6Rru3HkdlS
ktc4n3A+Hik253uJHaAPghxHuu4QNgH//Q9Py8avLx7Me36Fggkcg8tcP6rlgDH+sHT2OxqG+tVJ9OxsOidv0a7Xez6H0Nmz9JYulSUGflo5SJcUscwOHjxnhlw/u
```

After Decryption, image gets back in original format –

- **Project 2 – Encryption & Decryption using RSA Algorithm**

Java file attached.

```java
import java.math.*;

public class RSA_Algorithm_Encrypt_Decrypt {

    static int gcd(int e, int z)
        {
            if (e == 0)
                return z;
            else
                return gcd(z % e, e);
        }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
         int p, q, n, phai_n, d = 0, e, i;

         Scanner sc = new Scanner (System.in);

            int msg = 222;
            double cipher;
            BigInteger msgback;

            System.out.println("Enter 1st prime number p: ");
            p = sc.nextInt();

            System.out.println("Enter 2nd prime number q: ");
            q = sc.nextInt();

            // Calculating n
            n = p * q;

            // Calculating phai_n
            phai_n = (p - 1) * (q - 1);
            System.out.println("the value of phai_n = " + phai_n);

            for (e = 2; e < phai_n; e++) {

                // e is for public key exponent
                if (gcd(e, phai_n) == 1) {
                    break;
                }
            }

            System.out.println("the value of e = " + e);

            for (i = 0; i <= 9; i++) {
                int x = 1 + (i * phai_n);

                // d is for private key exponent
                if (x % e == 0) {
                    d = x / e;
                    break;
                }
            }
            System.out.println("the value of d = " + d);
            cipher = (Math.pow(msg, e)) % n;

            System.out.println("Encrypted message is : " + cipher);

            // converting int value of n to BigInteger which helps in
            // very big integer calculations that are outside the limit of all available primitive data types
            BigInteger N = BigInteger.valueOf(n);

            // converting float value of c to BigInteger
            BigInteger C = BigDecimal.valueOf(cipher).toBigInteger();
            msgback = (C.pow(d)).mod(N);
            System.out.println("Decrypted message is : "
                               + msgback);
        }

}
```

Output –

```
<terminated> RSA_Algorithm [Java Application] C:\Program Files\Java\jdk-15.0.
Enter 1st prime number p:
3
Enter 2nd prime number q:
7
the value of phai_n = 12
the value of e = 5
the value of d = 5
Encrypted message is : 3.0
Decrypted message is : 12
```

- **Project 3 – Implementation of Diffie-Hellman Algorithm**

Java file attached.

```java
import java.util.*;

public class Diffie_Hellman_Algo_Implementation {

    private static long cal_Power(long m, long n, long P)
       {
           long result = 0;
           if (n == 1){
               return m;
           }
           else{
               result = ((long)Math.pow(m, n)) % P;
               return result;
           }
       }


    public static void main(String[] args) {
        // TODO Auto-generated method stub
        long P, G, A, a, B, b, ka, kb;

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter value for public key G:");
        G = sc.nextLong();

        System.out.println("Enter value for public key P:");
        P = sc.nextLong();

        // Input from user for private keys a and b selected by User1 and User2
        System.out.println("Enter value for private key a selected by user1:");
        a = sc.nextLong();
        System.out.println("Enter value for private key b selected by user2:");
        b = sc.nextLong();

        // call calculatePower() method to generate A and B keys
        A = cal_Power(G, a, P);
        B = cal_Power(G, b, P);
        // call calculatePower() method to generate ka and kb secret keys after the exchange of x and y keys
        // calculate secret key for User1
        ka = cal_Power(B, a, P);
        // calculate secret key for User2
        kb = cal_Power(A, b, P);
        // print s3ecret keys of user1 and user2
        System.out.println("Secret key for User1 is:" + ka);
        System.out.println("Secret key for User2 is:" + kb);

    }
```

Output –

```
<terminated> Diffie_Hellman_Algo_Implementation [Java Application] C:\Progra
Enter value for public key G:
43
Enter value for public key P:
15
Enter value for private key a selected by user1:
6
Enter value for private key b selected by user2:
4
Secret key for User1 is:1
Secret key for User2 is:1
```