**Name: Gondaliya Krish D. Practical: 1**

**Roll no: 19BCE072 Date: 28/08/2022**

**Course: Blockchain Technology (2CS701)**

**Practical 1**

**Aim:** To implement digital signature to sign and verify authenticated user. Also, show a message when tampering is detected.

**Code**

import math

class RSA:

    def \_\_init\_\_(self,p=353,q=313):

        self.p=p

        self.q=q

        self.n=self.p\*self.q

        self.e=self.Public\_key()

        self.d=self.Private\_key()

        self.public\_key={'e':self.e,'n':self.n}

        self.private\_key={'d':self.d,'n':self.n}

    def set\_p(self,p):

        self.p=p

    def set\_q(self,q):

        self.q=q

    def GCD(self,n1,n2):

        if n2==0:

            return n1

        else:

            return self.GCD(n2,n1%n2)

    def Public\_key(self):

        phi=(self.p-1)\*(self.q-1)

        for i in range(2,phi):

            if(self.GCD(phi,i)==1):

                e=i

                break

        else:

            e=phi-1

        return e

    def Private\_key(self):

        phi=(self.p-1)\*(self.q-1)

        for i in range(2,phi):

            if((self.e\*i)%phi==1):

                d=i

                break

            else:

                d=phi-1

        return d

    def Encryption(self,C):

        return(math.ceil(C\*\*self.public\_key['e']%self.public\_key['n']))

    def Decryption(self,C):

        return(math.ceil(C\*\*self.private\_key['d']%self.private\_key['n']))

    def check\_message(self,sended,recived):

        if(sended==recived):

            print("\n"+"-"\*50)

            print(" ----> Message transfered without any tempering -----")

            print("-"\*50)

        else:

            print("\n"+"-"\*55)

            print(" Error : ----> Message transfered with some tempering -----")

            print("-"\*55)

if \_\_name\_\_=="\_\_main\_\_":

    r=RSA()

    send\_message=500

    print(" ----- Case 1 : Without any tampering -----\n")

    print("Public Key : ",r.public\_key)

    print("Private Key : ",r.private\_key)

    print("Orignal Key : ",send\_message)

    encrypted\_message=r.Encryption(send\_message)

    decryped\_message=r.Decryption(encrypted\_message)

    print("Encreypted messagesage : ",encrypted\_message)

    print("Decreypted messagesage : ",decryped\_message)

    r.check\_message(send\_message,decryped\_message)

    print(" ----- Case 2 : After tampering Public Key -----\n")

    r.public\_key['e']=9

    print("Public Key : ",r.public\_key)

    print("Private Key : ",r.private\_key)

    print("Orignal Key : ",send\_message)

    encrypted\_message=r.Encryption(send\_message)

    decryped\_message=r.Decryption(encrypted\_message)

    print("Encreypted messagesage : ",encrypted\_message)

    print("Decreypted messagesage : ",decryped\_message)

    r.check\_message(send\_message,decryped\_message)

    print("----- Case 3 : After tamper Encrypted data ----- \n")

    r.public\_key['e']=r.e

    print("Public Key : ",r.public\_key)

    print("Private Key : ",r.private\_key)

    print("Orignal Key : ",send\_message)

    encrypted\_message=r.Encryption(send\_message)

    decryped\_message=r.Decryption(encrypted\_message+1000)

    print("Encreypted messagesage : ",encrypted\_message)

    print("Decreypted messagesage : ",decryped\_message)

    r.check\_message(send\_message,decryped\_message)

**Output:**

