Digital assignment

Secure payment system

code:-

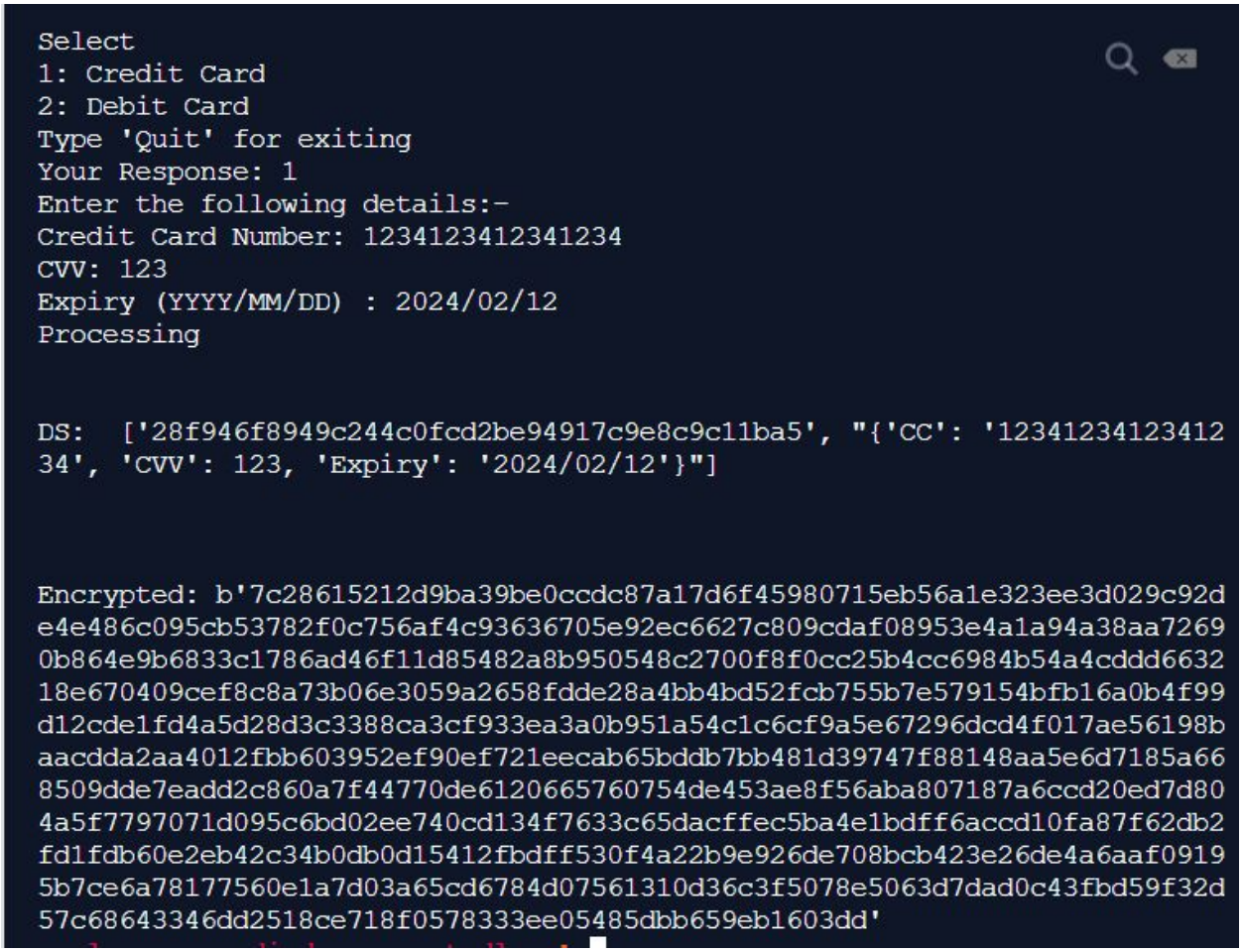
```
import time
import datetime as dt


from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
import binascii

keyPair = RSA.generate(3072)

pubKey = keyPair.publickey()
pubKeyPEM = pubKey.exportKey()
privKeyPEM = keyPair.exportKey()

def validate(date_text):
    try:
        dt.datetime.strptime(date_text, '%Y/%m/%d')
    except Exception:
        exit("Incorrect data format, should be YYYY/MM/DD")

def processing():
    print("Processing")



def ask(value):
    print("Enter the following details:-")
    cc = str(input(f"{value} Number: "))
    if (len(cc) != 16):
        exit(f"[-] Invalid {value} Number")
    cvv = int(input("CVV: "))
    if cvv not in range(100, 1000):
        exit("[-] Invalid CVV")
    expiry = str(input("Expiry (YYYY/MM/DD) : "))
    validate(expiry)
    processing()
```

```
    lst = {}
    lst["CC"]=cc
    lst["CVV"]=cvv
    lst["Expiry"]=expiry

    return lst


def sha1(data):
    bytes = ""

    h0 = 0x67452301
    h1 = 0xEFCDAB89
    h2 = 0x98BADCFE
    h3 = 0x10325476
    h4 = 0xC3D2E1F0

    for n in range(len(data)):
        bytes += '{0:08b}'.format(ord(data[n]))
    bits = bytes + "1"
    pBits = bits
    while len(pBits) % 512 != 448:
        pBits += "0"
    pBits += '{0:064b}'.format(len(bits) - 1)

    def chunks(l, n):
        return [l[i:i + n] for i in range(0, len(l), n)]

    def rol(n, b):
        return ((n << b) | (n >> (32 - b))) & 0xffffffff

    for c in chunks(pBits, 512):
        words = chunks(c, 32)
        w = [0] * 80
        for n in range(0, 16):
            w[n] = int(words[n], 2)
        for i in range(16, 80):
            w[i] = rol((w[i - 3] ^ w[i - 8] ^ w[i - 14] ^ w[i - 16]), 1)

        a = h0
        b = h1
        c = h2
        d = h3
```

```
        e = h4

        for i in range(0, 80):
            if 0 <= i <= 19:
                f = (b & c) | ((~b) & d)
                k = 0x5A827999
            elif 20 <= i <= 39:
                f = b ^ c ^ d
                k = 0x6ED9EBA1
            elif 40 <= i <= 59:
                f = (b & c) | (b & d) | (c & d)
                k = 0x8F1BBCDC
            elif 60 <= i <= 79:
                f = b ^ c ^ d
                k = 0xCA62C1D6

            temp = rol(a, 5) + f + e + k + w[i] & 0xffffffff
            e = d
            d = c
            c = rol(b, 30)
            b = a
            a = temp

        h0 = h0 + a & 0xffffffff
        h1 = h1 + b & 0xffffffff
        h2 = h2 + c & 0xffffffff
        h3 = h3 + d & 0xffffffff
        h4 = h4 + e & 0xffffffff

    return '%08x%08x%08x%08x%08x' % (h0, h1, h2, h3, h4)

def genDS(value):
    digitalSignature = []
    special = ["/", ",", "{", "}"]
    val = value
    val2 = ""
    for i in val:
        if i in special:
            val2 += "A"
            continue
        val2 += i

    val3 = sha1(val2)
```

```
    digitalSignature.append(str(val3))
    digitalSignature.append(str(val))
    print("\n\nDS: ", digitalSignature)

    return str(digitalSignature)

print("Select ")
print("1: Credit Card")
print("2: Debit Card")
print("Type 'Quit' for exiting")
response =""
try:
    while(response!="quit"):
        response = str(input("Your Response: ")).lower()

        if(response=="1"):
            enc_msg333=genDS(ask("Credit Card"))
            msg = b'enc_msg333'
            encryptor = PKCS1_OAEP.new(pubKey)
            encrypted = encryptor.encrypt(msg)
            print("\n\n")
            print("Encrypted:", binascii.hexlify(encrypted))
            exit()
        elif(response=="2"):
            enc_msg333=genDS(ask("Debit Card"))
            msg = b'enc_msg333'
            encryptor = PKCS1_OAEP.new(pubKey)
            encrypted = encryptor.encrypt(msg)
            print("\n\n")
            print("Encrypted:", binascii.hexlify(encrypted))
            exit()


except FileNotFoundError:
    print("Invalid Input")
    exit()
```

Input:-



- First, take the input
- all the input will save as a list
- Then the whole list will be encrypted as RSA

AENA VERMA
19BCI0221