

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:-

```
Select
1: Credit Card
2: Debit Card
Type 'Quit' for exiting
Your Response: 1
Enter the following details:-
Credit Card Number: 1234123412341234
CVV: 123
Expiry (YYYY/MM/DD) : 2024/02/12
Processing

DS:  ['28f946f8949c244c0fcd2be94917c9e8c9c11ba5', '{"CC': '1234123412341234', 'CVV': 123, 'Expiry': '2024/02/12'}"]

Encrypted: b'7c28615212d9ba39be0ccdc87a17d6f45980715eb56a1e323ee3d029c92de4e486c095cb53782f0c756af4c93636705e92ec6627c809cdaf08953e4a1a94a38aa72690b864e9b6833c1786ad46f11d85482a8b950548c2700f8f0cc25b4cc6984b54a4cddd663218e670409cef8c8a73b06e3059a2658fdde28a4bb4bd52fcb755b7e579154bfb16a0b4f99d12cde1fd4a5d28d3c3388ca3cf933ea3a0b951a54c1c6cf9a5e67296dcd4f017ae56198baacdda2aa4012fbb603952ef90ef721eecab65bddb7bb481d39747f88148aa5e6d7185a668509dde7eadd2c860a7f44770de6120665760754de453ae8f56aba807187a6ccd20ed7d804a5f7797071d095c6bd02ee740cd134f7633c65dacffec5ba4e1bdf6accd10fa87f62db2fd1fdb60e2eb42c34b0db0d15412fbdff530f4a22b9e926de708bcb423e26de4a6aaf09195b7ce6a78177560e1a7d03a65cd6784d07561310d36c3f5078e5063d7dad0c43fbd59f32d57c68643346dd2518ce718f0578333ee05485dbb659eb1603dd'
```

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

AENA VERMA

19BCI0221