

Secure Payment System

I've Created this program by using RSA,SHA-1 and also tried securing the program by Input Validation. Crashing the program is next to impossible and also this program can be implemented in Real World Scenarios.
Doing this assignment was fun.

Hritish Kumar

Output:



Select the payment System

1: Credit Card

2: Debit Card

3: UPI

Type 'Quit' for exit

Your Response: 1

Enter the following details:-

Credit Card Number: 1234123412341234

CVV: 123

Expiry (YYYY/MM/DD) : 2019/10/11

[+] Payment Processing

[+] Generating the Hashes

[+] This process may take a while....

Digital Signature: ['28f946f8949c244c0fcd2be94917c9e8c9c11ba5', '{"CC': '1234123412341234', 'CVV': 123, 'Expiry': '2019/10/11'}"]

Encrypted: b'0df09ea4b8e837db849d71862c924696ae8b0c5caca9cdde0c14517d38e9550da151c7579169451ce6b445735a06d609a0b2f720e2b54e3e5f93f59476d94f8dbdd97c49c17e934c15f308c69ede27ef1cd0e7300d8e02e11a8642e4fff220651e311b0274e9498d93a18ab87750437e32c29326c207ed304bbf00c81394c95cca22b9c647612e56a19726d69cdf2fed59f02b8da064211fdb8ce6a057472836d2f01a4f795e9814b611fe143f74f67cf21d3335840df5f941a705839b0614a334d1ac09172e58f260dc1d85d2a2e8444779017113a74e8f43718ef680f491cb16ff5f0478f9ealae755644a39401418b2c7ff0029f17e5c5843a97505286d4923bfd4d8bbdc8ed4d50b2105289e1f69e6967079c517b9b7c4930e8b1ec5410a4e4ad3ff76ae8781b2d6167ce21bda1307544d2c434e0dadf9dfb9ebe0a450c3cbfffb0e9f196702ce926cd838b8c983fb48da4750006f036a5c4c09a25a777e6b4e774e2daec726841a2cb7c46c29425a8c4e0ab8662fc047fb689828d43153e'

[+] Payment Successfull

repl process died unexpectedly: exit status 1

OVERVIEW:

1. The program takes basic inputs.
2. Generates hashes of all the inputs.
3. Creates a Digital Signature.
4. Now, uses RSA to encrypt the Digital signature.
5. Uncommenting some lines will give verbose Output.

Input:

Just enter realistic details of cc or debit card.

Or

Check ScreenShot

Code:-

#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()

#print(f"Public key: (n={hex(pubKey.n)}, e={hex(pubKey.e)})")

pubKeyPEM = pubKey.exportKey()

#print(pubKeyPEM.decode('ascii'))

#print(f"Private key: (n={hex(pubKey.n)}, d={hex(keyPair.d)})")

privKeyPEM = keyPair.exportKey()

#print(privKeyPEM.decode('ascii'))

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("[+] Payment Processing ")

time.sleep(3)

print("[+] Generating the Hashes")

```
time.sleep(3)
print("[+] This process may take a while....")
time.sleep(3)
```

```
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\nDigital Signature: ", digitalSignature)

    return str(digitalSignature)
```

```
print("""
```

""""")

```
print("Select the payment System")
print("1: Credit Card")
print("2: Debit Card")
print("3: UPI")
print("Type 'Quit' for exit")
```

```

response = ""
try:
    while(response!="quit"):
        response = str(input("Your Response: ")).lower()

        #For Credit Card
        if(response=="1"):
            enc_msg=genDS(ask("Credit Card"))
            msg = b'enc_msg'
            encryptor = PKCS1_OAEP.new(pubKey)
            encrypted = encryptor.encrypt(msg)
            print("\n\n")
            print("Encrypted:", binascii.hexlify(encrypted))
            exit("[+] Payment Successfull")
        elif(response=="2"):
            enc_msg=genDS(ask("Debit Card"))
            msg = b'enc_msg'
            encryptor = PKCS1_OAEP.new(pubKey)
            encrypted = encryptor.encrypt(msg)
            print("\n\n")
            print("Encrypted:", binascii.hexlify(encrypted))
            exit("[+] Payment Successfull")

        #Under Construction Try using Credit Card or Debit Card Methods
        elif(response=="3"):
            exit("Under Construction \nTry using Credit Card or Debit Card Method ")
            upi_id=str(input("Enter the UPI id: "))
            if("@" not in upi_id):
                exit("[-] Invalid Upi Id")
            processing()

except FileNotFoundError:
    print("[-] Invalid Input")
    exit("[-] Exiting Program")

```

=====

The screenshot shows a web browser with a Python script titled 'main.py' being executed. The script is a simple payment system that prompts the user for a response. If the user enters 'quit', the program exits. If the user enters '1', it prompts for a credit card number and processes the payment. The script uses pycryptodome for encryption and decryption. The terminal output shows the script running successfully, displaying the encrypted card number and the payment status.

```
main.py
154 print("2: Debit Card")
155 print("3: UPI")
156 print("Type 'Quit' for exit")
157 response = ""
158 try:
159     while(response!="Quit"):
160         response = str(input("Your Response: ")).lower()
161
162         #For Credit Card
163         if(response=="1"):
164             enc_msg=genDS(ask("Credit Card"))
165             msg = b'enc_msg'
166             encryptor = PKCS1_OAEP.new(pubKey)
167             encrypted = encryptor.encrypt(msg)
168             print("\n\n")
169             print("Encrypted:", binascii.hexlify(encrypted))
170             exit("\n Payment Successful")
171         elif(response=="2"):
172             enc_msg=genDS(ask("Debit Card"))
173             msg = b'enc_msg'
174             encryptor = PKCS1_OAEP.new(pubKey)
175             encrypted = encryptor.encrypt(msg)
176             print("\n\n")
177             print("Encrypted:", binascii.hexlify(encrypted))
178             exit("\n Payment Successful")
179
180 #Under Construction Try using Credit Card or Debit Card Methods
181 elif(response=="3"):
182     exit("Under Construction \nTry using Credit Card or Debit Card Method ")
183     upi_id=str(input("Enter the UPI id: "))
184     if "@" not in upi_id:
185         exit("\n Invalid Upi id")
186     processing()
187
188
189 except FileNotFoundError:
190     print("\n Invalid Input")
191     exit("\n Exiting Program")
192
193
```

The terminal output shows the script running successfully, displaying the encrypted card number and the payment status.

```
Select the Payment System
1: Credit Card
2: Debit Card
3: UPI
Type 'Quit' for exit
Your Response: 1
Enter the following details:-
Credit Card Number: 1234123412341234
CVV: 123
Expiry (MM/DD/YYYY): 2019/10/11
[+] Payment Processing
[+] Generating the Hashes
[+] This process may take a while....

Digital Signature: ["28E94669949c2440ef0d3be94910e9e801ba5", "1cc": "1234123412341234",
"CVV": 123, "Expiry": "2019/10/11"]

Encrypted: b'0df09ea4bde837db849d718c2c924696ae8b0c30acae8dcde0c4517d38e9550da151c7579169451c
e8b45f72bae6d0b0b2f1220a24e3e6318a9f76168d8d97c9b27e93a15130c0f0e2b274c0e07200e0
2e1a18642e4f22063e1311b027e49498993a18ab8761053b7432c9326c207ed304bbf00b01394e950ca22b9c6476
12656a19726d69d72ed9f02b08d064211fdba80e6a574728362d2f01a1795e9814b1611e4374c7c721d333
58f0e5c7e17038390e41833401e09272e3f240b01583b24447907711a7e8437f1ed0e04912eb44
56f0479eae1a75564a334041182c7f7f0029f17e5c843a1975052864923bdf48bbdc8d4d30b2105289e1f69e
6967079c17b9b74930eb1e0c410e4ed34f76e871b2d61c7c0e21bda1307544d2c434d0adcf9df9e8e0a450
c8bdf20e0c21870102e9286c380e370b40de75006e0304e5c0492a5e7776eb4e77fdeade72e641a2b7e466
2942583e0e0ab8662f6047f5b6982824b3153e'
[+] Payment Successful
repl process died unexpectedly: exit status 19
```




Select the payment System

1: Credit Card

2: Debit Card

3: UPI

Type 'Quit' for exit

Your Response: 1

Enter the following details:-

Credit Card Number: 1234123412341234

CVV: 123

Expiry (YYYY/MM/DD) : 2019/10/11

[+] Payment Processing

[+] Generating the Hashes

[+] This process may take a while....

Digital Signature: ['28f946f8949c244c0fcd2be94917c9e8c9c11ba5', '{"CC': '1234123412341234', 'CVV': 123, 'Expiry': '2019/10/11'}"]

Encrypted: b'0df09ea4b8e837db849d71862c924696ae8b0c5caca9cdde0c14517d38e9550da151c7579169451ce6b445735a06d609a0b2f720e2b54e3e5f93f59476d94f8dbdd97c49c17e934c15f308c69ede27ef1cd0e7300d8e02e11a8642e4ff220651e311b0274e9498d93a18ab87750437e32c29326c207ed304bbf00c81394c95cca22b9c647612e56a19726d69cdf2fed59f02b8da064211fdbba8ce6a057472836d2f01a4f795e9814b611fe143f74f67cf21d3335840df5f941a705839b0614a334d1ac09172e58f260dc1d85d2a2e8444779017113a74e8f43718ef680f491cb16ff5f0478f9ea1ae755644a39401418b2c7ff0029f17e5c5843a97505286d4923bfd4d8bbdc8ed4d50b2105289e1f69e6967079c517b9b7c4930e8b1ec5410a4e4ad3ff76ae8781b2d6167ce21bda1307544d2c434e0dadf9dfb9ebe0a450c3cbfffb0e9f196702ce926cd838b8c983fb48da4750006f036a5c4c09a25a777e6b4e774e2daec726841a2cb7c46c29425a8c4e0ab8662fc047fb689828d43153e'

[+] Payment Successfull

repl process died unexpectedly: exit status 1 ✖

Hritish Kumar