UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

DEPARTAMENTO DE PROGRAMACIÓN DE COMPUTADORAS

LICENCIATURA EN CIBERSEGURIDAD

CRIPTOGRAFÍA

LABORATORIO#4

PERTENECE A:

POLANCO, OMAR 8-1014-120

PROFESOR:

MORENO, JOSE

GRUPO:

1S3121

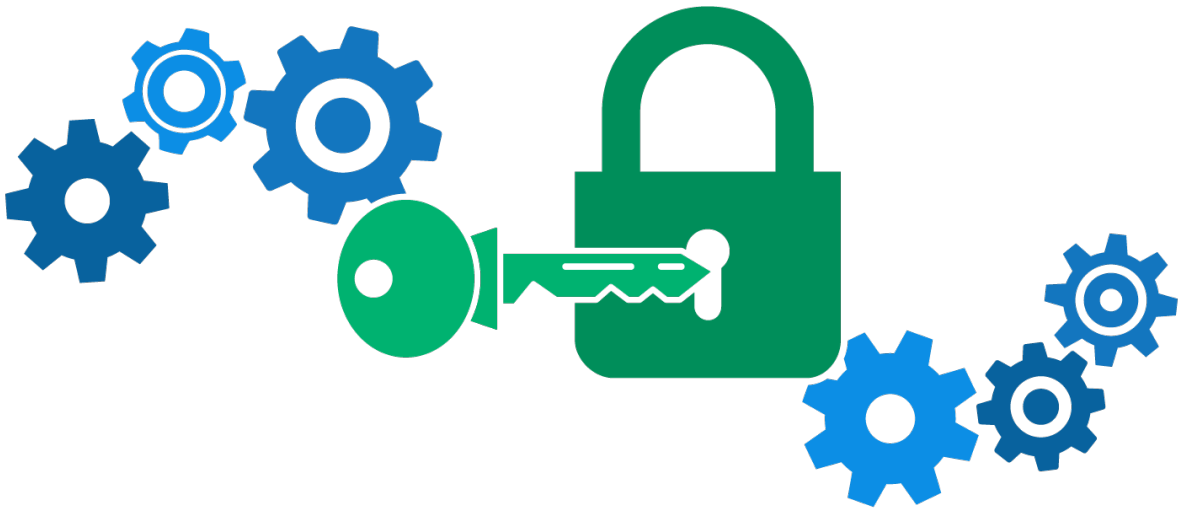
II SEMESTRE

2024

Índice

Introducción

En el ámbito de la ciberseguridad, uno de los aspectos fundamentales es el manejo adecuado de las claves criptográficas, ya que son esenciales para garantizar la seguridad de la información. El presente informe tiene como objetivo desarrollar un enfoque práctico para extraer la clave privada \(d \) de un archivo PEM que contiene una clave RSA. Para este fin, se utilizará principalmente la herramienta OpenSSL, que es ampliamente reconocida en la industria para la gestión de claves criptográficas. Adicionalmente, se explorará el uso de la librería PyCryptodome en Python, la cual permitirá la manipulación programática de las claves RSA, asegurando la correcta extracción del exponente \(d \) y su conversión a una representación decimal adecuada. Este proceso no solo permitirá una comprensión más profunda del funcionamiento de las claves privadas en RSA, sino que también será una demostración práctica de cómo emplear herramientas y librerías de código abierto en la seguridad informática.



Lectura del archivo PEM

Primero debes de descargar el archivo PEM y al presenciar que contiene, observamos que tiene una llave privada RSA

```
omars@MacBook-Pro-3 Downloads % cat privacy_enhanced_mail_1f696c053d
76a78c2c531bb013a92d4a.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAzvKDt+EO+A6oE1LItSunkWJ8vN6Tgcu8Ck077joGDfG2NtxD
4vyQxGTQngr6jEKJuVz2MIwDcdXtFLIF+ISX9HfALQ3yiedNS80n/TR1BNcJSlzI
uqLmFxddmjmfUvHFuFLvxgXRga3mg3r7olTW+1fxOS0ZVeDJqFCaORRvoAYOgLgu
d2/E0aaaJi9cN7CjmdJ7Q3m6ryGuCwqEvZ1KgVWWa7fKcFopnl/fcsSecwbDV5hW
fmvxiAUJy1mNSPwkf5YhGQ+83g9N588RpLLMXmgt6KimtiWnJsqtDPRlY4Bjxdpu
V3QyUdo2ymqnquZnE/vlU/hn6/s8+ctdTqfSCwIDAQABAoIBAHw7HVNPKZtDwSYI
djA8CpW+F7+Rpd8vHKzafHWgI25PgeEhDSfAEm+zTYDyekGk1+SMp8Ww54h4sZ/Q
1sC/aDD7ikQBsW2TitVMTQs1aGIFbLBVTrKrg5CtGCWzHa+/L8BdGU84wvIkINMh
CtoCMCQmQMrgBeuFy8jcyhgl6nSW2bFwxcv+NU/hmmMQK4LzjV18JRc1IIuDpUJA
kn+JmEjBal/nDOlQ2v97+fS3G1mBAaUgSM0wwWy5lDMLEFktLJXU0OV59Sh/90qI
Jo0DiWmMj3ua6BPzkkaJPQJmHPCNnLzsn3Is920OlvHhdzfins6GdnZ8tuHfDb0t
cx7YSLECgYEA7ftHFeupO8TCy+cSyAgQJ8yGqNKNLHjJcg5t5vaAMeDjT/pe7w/R
0IWuScCoADiL9+6YqUp34RgeYDkks7O7nc6XuABi8oMMjxGYPfrdVfH5zlNimS4U
wl93bvfazutxnhz58vYvS6bQA95NQn7rWk2YFWRPzhJVkxvfK6N/x6cCgYEA3p21
w10lYvHNNiI0KBjHvroDMyB+39vD8mSObRQQuJFJdKWuMq+o5OrkC0KtpYZ+Gw4z
L9DQosip3hrb7b2B+bq0yP7Izj5mAVXizQTGkluT/YivvgXcxVKoNuNTqTEgmyOh
Pn6w+PqRnESsSFzjfWrahTCrVomcZmnUTFh0rv0CgYBETN68+tKqNbFWhe4M/Mtu
MLPhFfSwc8YU9vEx3UMzjYCPvqKqZ9bmyscXobRVw+Tf9llYFOhM8Pge06el74qE
IvvGMk4zncrn8LvJ5grKFNWGEsZ0ghYxJucHMRlaU5ZbM6PEyEUQqEKBKbbww65W
T3i7gvuof/iRbOljA9yzdwKBgQDT9Pc+Fu7k4XNRCon8b3OnnjYztMn4XKeZn7KY
GtW81eBJpwJQEj5OD3OnYQoyovZozkFgUoKDq2lJJuul1ZzuaJ1/Dk+lR3YZ6Wtz
ZwumCHnEmSMzWyOT4Rp2gEWEv1jbPbZl6XyY4wJG9n/OulqDbHy4+dj5ITb/r93J
/yLCBQKBgHa8XYMLzH63Ieh69VZF/7jO3d3lZ4LlMEYT0BF7synfe9q6x7s0ia9b
f6/QCkmOxPC868qhOMgSS48L+TMKmQNQSm9b9oy2ILlLA0KDsX5O/Foyiz1scwr7
nh6tZ+tVQCRvFviIEGkaXdEiBN4eTbcjfc5md/u9eA5N21Pzgd/G
-----END RSA PRIVATE KEY-----
```

Extracción de la clave privada (d)

Una vez cargada la clave, se podrá acceder a la clave privada (d) como un entero decimal.

Ahora con el siguiente comando podremos observar la clave.

Openssl rsa -noout -text -inform PEM -in key.pem -pubout

```
omars@MacBook-Pro-3 Downloads % openssl rsa -in privacy_enhanced_mail_1f69
6c053d76a78c2c531bb013a92d4a.pem -noout -text
```

Lo que nos da lo siguientes valores de la clave.

```
privateExponent:
    7c:3b:1d:53:4f:29:9b:43:c1:26:08:76:30:3c:0a:
    95:be:17:bf:91:a5:df:2f:1c:ac:da:7c:75:a0:23:
    6e:4f:81:e1:21:0d:27:c0:12:6f:b3:4d:80:f2:7a:
    41:a4:d7:e4:8c:a7:c5:b0:e7:88:78:b1:9f:d0:d6:
    c0:bf:68:30:fb:8a:44:01:b1:6d:93:8a:d5:4c:4d:
    0b:35:68:62:05:6c:b0:55:4e:b2:ab:83:90:ad:18:
    25:b3:1d:af:bf:2f:c0:5d:19:4f:38:c2:f2:24:20:
    d3:21:0a:da:02:30:24:26:40:ca:e0:05:eb:85:cb:
    c8:dc:ca:18:25:ea:74:96:d9:b1:70:c5:cb:fe:35:
    4f:e1:9a:63:10:2b:82:f3:8d:5d:7c:25:17:35:20:
    8b:83:a5:42:40:92:7f:89:98:48:c1:6a:5f:e7:0c:
    e9:50:da:ff:7b:f9:f4:b7:1b:59:81:01:a5:20:48:
    cd:30:c1:6c:b9:94:33:0b:10:59:2d:2c:95:d4:d0:
    e5:79:f5:28:7f:f7:4a:88:26:8d:03:89:69:8c:8f:
    7b:9a:e8:13:f3:92:46:89:3d:02:66:1c:f0:8d:9c:
    bc:ec:9f:72:2c:f7:6d:0e:96:f1:e1:77:37:e2:9e:
    ce:86:76:76:7c:b6:e1:df:0d:bd:2d:73:1e:d8:48:
    b1
```

Visualización y manipulación de la clave (d)

Primero debemos eliminar los (:) y convertir todas las minusculas en mayusculas, copiamos los valores de la clave privada y corremos el siguiente comando:

echo 'clave privada | sed -e 's/://g' | tr 'a-z' 'A-Z'

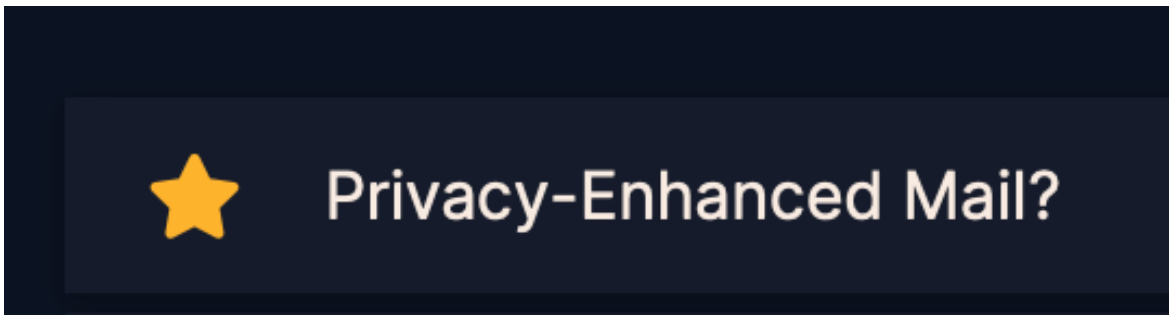
```
omars@MacBook-Pro-3 Downloads % echo '7c:3b:1d:53:4f:29:9b:43:c1:26:
08:76:30:3c:0a:95:be:17:bf:91:a5:df:2f:1c:ac:da:7c:75:a0:23:6e:4f:81
:e1:21:0d:27:c0:12:6f:b3:4d:80:f2:7a:41:a4:d7:e4:8c:a7:c5:b0:e7:88:7
8:b1:9f:d0:d6:c0:bf:68:30:fb:8a:44:01:b1:6d:93:8a:d5:4c:4d:0b:35:68:
62:05:6c:b0:55:4e:b2:ab:83:90:ad:18:25:b3:1d:af:bf:2f:c0:5d:19:4f:38
:c2:f2:24:20:d3:21:0a:da:02:30:24:26:40:ca:e0:05:eb:85:cb:c8:dc:ca:1
8:25:ea:74:96:d9:b1:70:c5:cb:fe:35:4f:e1:9a:63:10:2b:82:f3:8d:5d:7c:
25:17:35:20:8b:83:a5:42:40:92:7f:89:98:48:c1:6a:5f:e7:0c:e9:50:da:ff
:7b:f9:f4:b7:1b:59:81:01:a5:20:48:cd:30:c1:6c:b9:94:33:0b:10:59:2d:2
c:95:d4:d0:e5:79:f5:28:7f:f7:4a:88:26:8d:03:89:69:8c:8f:7b:9a:e8:13:
f3:92:46:89:3d:02:66:1c:f0:8d:9c:bc:ec:9f:72:2c:f7:6d:0e:96:f1:e1:77
:37:e2:9e:ce:86:76:76:7c:b6:e1:df:0d:bd:2d:73:1e:d8:48:b1' | sed -e
's/://g' | tr 'a-z' 'A-Z'
7C3B1D534F299B43C1260876303C0A95BE17BF91A5DF2F1CACDA7C75A0236E4F81E1
210D27C0126FB34D80F27A41A4D7E48CA7C5B0E78878B19FD0D6C0BF6830FB8A4401
B16D938AD54C4D0B356862056CB0554EB2AB8390AD1825B31DAFBF2FC05D194F38C2
F22420D3210ADA0230242640CAE005EB85CBC8DCCA1825EA7496D9B170C5CBFE354F
E19A63102B82F38D5D7C251735208B83A54240927F899848C16A5FE70CE950DAFF7B
F9F4B71B598101A52048CD30C16CB994330B10592D2C95D4D0E579F5287FF74A8826
8D0389698C8F7B9AE813F39246893D02661CF08D9CBCEC9F722CF76D0E96F1E17737
E29ECE8676767CB6E1DF0DBD2D731ED848B1
```

Y ahora, convertiremos la clave privada (d) en su representación decimal y mostrarla, con el siguiente comando:

echo "ibase=16; clave privada" | bc

```
omars@MacBook-Pro-3 Downloads % echo "ibase=16; 7C3B1D534F299B43C126
0876303C0A95BE17BF91A5DF2F1CACDA7C75A0236E4F81E1210D27C0126FB34D80F2
7A41A4D7E48CA7C5B0E78878B19FD0D6C0BF6830FB8A4401B16D938AD54C4D0B3568
62056CB0554EB2AB8390AD1825B31DAFBF2FC05D194F38C2F22420D3210ADA023024
2640CAE005EB85CBC8DCCA1825EA7496D9B170C5CBFE354FE19A63102B82F38D5D7C
251735208B83A54240927F899848C16A5FE70CE950DAFF7BF9F4B71B598101A52048
CD30C16CB994330B10592D2C95D4D0E579F5287FF74A88268D0389698C8F7B9AE813
F39246893D02661CF08D9CBCEC9F722CF76D0E96F1E17737E29ECE8676767CB6E1DF
0DBD2D731ED848B1" |bc
156827002880563313647871710458199736549911499491979599298608612281 80
\
021707316851924456205543665565810892674190059831330231436970914474 77
\
456271494562051914438978515890899418195134884601743250646416356496 09
\
937842541533954067991013147600334450651934295925123499520209829322 18
\
524462341002102063435489318813316464511621736943938440710470694912 33
\
623768021974620459512895916180059521636623753829644733537581887195 25
\
200269931021483288970835471842864932411915059536016688589411297909 66
\
909236941127851370202421135897091086763569884760099112291072056970 63
\
638041734901957976874805476010483879042470898826044392690667379597 51
\
04689
```

Introducimos el resultado en CryptoHack y ¡hemos terminado!

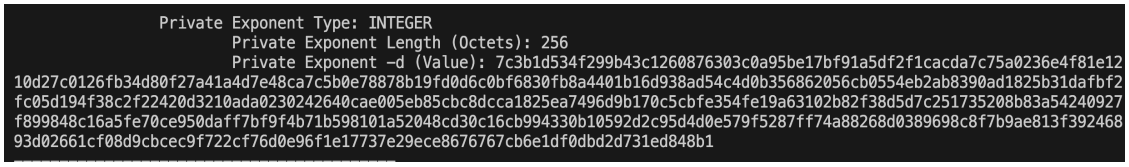


PyCryptodome (Python)

Otra forma mucho más sencilla que utilizar OpenSSL, tenemos dos opciones de programa, que nos deja ver todas claves de la llave privada que tenemos, el cual es el siguiente:

```
import base64

def pem_decoder(path_to_pem):
    with open(path_to_pem) as f:
        data = f.read()

    data = bytes(data[30:-30].encode("ascii"))
    hexData = base64.decodebytes(data).hex()

    ID = {
        "INTEGER":          0x02,
    }

    NAME = {}

    for i in ID:
        NAME[ID[i]] = i

    # An RSA private key shall have ASN.1 type RSAPrivateKey:
    #
    #    RSAPrivateKey ::= SEQUENCE {
    #      version Version,
    #      modulus INTEGER, -- n
    #      publicExponent INTEGER, -- e
    #      privateExponent INTEGER, -- d
    #      prime1 INTEGER, -- p
    #      prime2 INTEGER, -- q
    #      exponent1 INTEGER, -- d mod (p-1)
    #      exponent2 INTEGER, -- d mod (q-1)
    #      coefficient INTEGER -- (inverse of q) mod p }
    #
    #    Version ::= INTEGER

    typeArray = ["Version", "Modulus", "Public Exponent", "Private Exponent", "Prime 1", "Prime 2",
"Exponent 1",
                 "Exponent 2", "Coefficient"]

    lengthArray = ["(Octets)", "(Octets)", "(Hex)", "(Octets)", "(Octets)", "(Octets)", "(Octets)", "
(Octets)",
                   "(Octets)"]

    valueArray = ["", "-n", "-e", "-d", "-p", "-q", "- d mod (p-1)", "- d mod (q-1)", "- q^-1 mod p"]

    def lengthInOctets(x, y):
        return int(hexData[x:y], 16)

    print("\nConfirm length of hexData matches length value in ASN.1: " + str(len(hexData) ==
                                                                              (int(hexData[4:8], 16)*2)
+ 8))
    # The '+8' is to account for the 8 bits at the start that the ASN.1 length doesn't include
    print("-----------------------------------------------------------------------------------------
\n")

    x, y = 4, 8
    print("Sequence {")
    print("\tLength of Sequence: " + str(int(hexData[x:y], 16)*2))  # We can skip out bits 0-3 as we
know they will
    # be 3082

    x += 4
    y += 2

    for loop in range(0, 9):

        print("\t\t" + typeArray[loop] + " Type: " + NAME[int(hexData[x:y], 16) & 31])  # I confirm type
everytime as
        # more of a debug as it easily shows if we're in the wrong place

        x += 2
        y += 2
        if int(hexData[x:y], 16) == int(0x82):
            x += 2
            y += 4
            len_in_octs = lengthInOctets(x, y)

        elif int(hexData[x:y], 16) == int(0x81):

            x += 2
            y += 2
            len_in_octs = lengthInOctets(x, y)

        else:
            len_in_octs = lengthInOctets(x, y)

        print("\t\t\t" + typeArray[loop] + " Length " + lengthArray[loop] + ": " + str(len_in_octs))

        x = y
        y += len_in_octs * 2

        print("\t\t\t" + typeArray[loop] + " " + valueArray[loop] + " (Value): " + hexData[x:y])

        x = y
        y += 2

        print("-----------------------------------------\n")

    print("                                    }")


pem_decoder("privacy_enhanced_mail_1f696c053d76a78c2c531bb013a92d4a.pem")
```

Este programa nos deja apreciar todas las claves de la llave, pero solo nos interesa la clave privada.

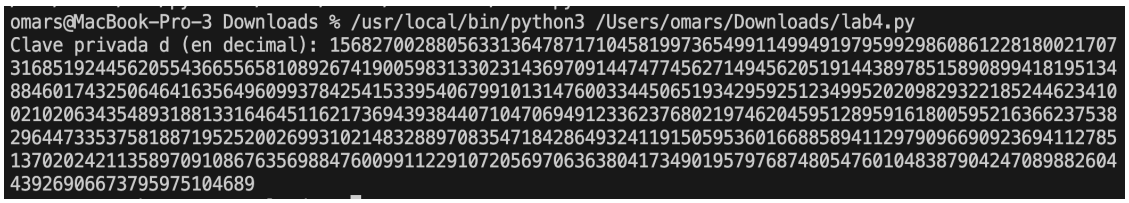


La cual es la misma que nos brindó OpenSSL.

También contamos el siguiente código:

```python
from Crypto.PublicKey import RSA
# Leer el archivo PEM
with open('privacy_enhanced_mail_1f696c053d76a78c2c531bb013a92d4a.pem', 'r') as f:
    key_data = f.read()
# Importar la clave RSA
rsa_key = RSA.importKey(key_data)
# Extraer el valor de la clave privada d
private_exponent_d = rsa_key.d
print(f'Clave privada d (en decimal): {private_exponent_d}')
```

El cual nos da únicamente la clave privada:



Conclusión

En conclusión, la extracción de la clave privada \(d \) de un archivo PEM que contiene una clave RSA es un proceso que puede realizarse eficazmente mediante el uso de herramientas como OpenSSL y librerías como PyCryptodome en Python. OpenSSL facilita la obtención de la clave en un formato estándar, mientras que PyCryptodome permite una manipulación más detallada y personalizada de las claves dentro de un entorno programático. A lo largo de este informe, se ha demostrado la importancia de comprender los detalles técnicos detrás de las claves criptográficas, ya que el exponente \(d \) es fundamental en los sistemas de cifrado RSA. Esta solución aporta un enfoque práctico para gestionar y manipular claves en diversos escenarios de seguridad informática.

Referencias

NSC. (23 de Nov de 2015). *Cracking the RSA keys (Part 1 – getting the private exponent)*. Obtenido de loginroot: https://loginroot.com/cracking-the-rsa-keys-part-1-getting-the-private-exponent/

Hollingworth, J. (10 de Jul de 2021). *What the F*@K is a Base64 DER ASN.1 (a.k.a. a PEM key)?* . Obtenido de Medium: https://jchhollingworth97.medium.com/what-the-f-k-is-a-base64-der-asn-1-a-k-a-a-pem-key-5800e13c4819

Extracción de clave privada (d) desde una clave RSA en formato PEM. (12 de Sep de 2024). *PDF*. Panamá, Ciudad de Panamá, Panamá.