INSTITUTO POLITÉCNICO NACIONAL

ESCOM

6-11-2024

Session 7: RSA schoolbook

Nombre: Álvarez García Brandon Azarael

Nombre de la materia: Selected topics in cryptography

Grupo: 7CM1

Nombre del profesor: Dra. Sandra Díaz Santiago

Ejercicios de programación

1. Design and implement a function to generate the pair of keys for RSA. Please consider the following requirements:
   - You must use a cryptographic pseudorandom function.
   - Each prime number must have 512 bits. You can use a function on a cryptographic library to generate each prime.
   - You can use a function in a library to compute a multiplicative inverse and/or calculate the greatest common divisor.
   - Your function must store the private key in base 64 in a text file. The public key must be stored, also in base 64 in a different text file.

Para esta parte cree una clase llamada RSA, en la cual se crearan los diferentes métodos que se solicitan a lo largo de la práctica, se separa en clases para tener un mejor control y orden en el código, yo me apoye de las librerías de Crypto de Python, para la generación de primos grandes

```
7   def key_generation(self):
8       bits = 512
9       p = number.getPrime(bits)
10      q = number.getPrime(bits)
11      n = p * q
12      z = (p-1) * (q-1)
13      k = 0
14
15
16      for i in range(2, z-1):
17
18          if number.GCD(z, i) == 1:
19              k = i
20              break
21
22      j = number.inverse(k, z)
23
24      #Base 64
25      n_base64 = base64.b64encode(n.to_bytes((n.bit_length() + 7) // 8, 'big')).decode('utf-8')
26      k_base64 = base64.b64encode(k.to_bytes((k.bit_length() + 7) // 8, 'big')).decode('utf-8')
27      j_base64 = base64.b64encode(j.to_bytes((j.bit_length() + 7) // 8, 'big')).decode('utf-8')
28
29      #Save in a file text
30      private_key = open("PrivateKey.txt", "w").write(j_base64)
31      public_key = open("PublicKey.txt", "w").write(n_base64+'\n'+k_base64)
32
```

Se crea el método de key_generation, el cual devuelve dos archivos de texto, uno incluye la llave publica y el otro la llave privada, y se hacen los cálculos correspondientes de los números primos (p y q) con la longitud de 512 bits, un valor k que sea coprimo con z y j es el calculo del inverso, posteriormente estos valores, se llevan a base 64 y se proceden a guardar en los archivos de texto generados

2. Design a function to encipher a message with RSA. Your function must receive a public key and the message. The output will be the ciphertext in base64.

En esta función se hace el calculo de el texto cifrado, en el cual recibimos un archivo de texto que contiene la llave publica y decodificamos estos valores de base 64 para manejarlos como enteros, se procede a realizar el calculo de cifrado y este resultado se vuelve a pasar a base 64 y esto es lo que retornaremos, el texto cifrado

```python
def cipher(self, publickey, message):

    publickey = open(publickey, "r").readlines()

    m = int.from_bytes(message.encode(), "big")
    n = int.from_bytes(base64.b64decode(publickey[0]), "big")
    k = int.from_bytes(base64.b64decode(publickey[1]), "big")

    C = (m**k) % n

    C_base64 = base64.b64encode(C.to_bytes((C.bit_length() + 7) // 8, 'big')).decode('utf-8')

    return C_base64
```

3. Design a function to decipher a ciphertext with RSA. Your function must receive a private key and a ciphertext. The output will be the plaintext.

En esta función realizamos el proceso contrario, obteniendo como entradas el archivo de texto donde contiene nuestra llave privada y el texto cifrado, también de la llave publica obtenemos el valor de n para el cálculo, realizamos dicho cálculo de descifrado y lo decodificamos para que pueda ser leído, esto es lo que retorna la función

```python
def decipher(self, privatekey, ciphertext):

    privatekey = open(privatekey, "r").readlines()
    publickey = open("PublicKey.txt", "r").readlines()

    j = int.from_bytes(base64.b64decode(privatekey[0]), "big")
    n = int.from_bytes(base64.b64decode(publickey[0]), "big")
    c = int.from_bytes(base64.b64decode(ciphertext), "big")

    M = pow(c, j, n)

    M_decoded = M.to_bytes((M.bit_length() + 7) // 8, 'big').decode('utf-8')

    return M_decoded
```

4. Use the previous functions to generate a key pair for RSA. Then encipher/decipher a password of at least eight characters. **Please avoid to edit your source code to change any parameter.**

Para este punto adicionalmente, se crea un método que genera una contraseña al azar entre letras, número y caracteres de longitud 8

```
1 usage
def password_generator(self):
    return ''.join(random.choice(string.ascii_letters + string.digits + string.punctuation) for _ in range(7))
```

Pruebas

Este es el archivo main, en donde simplemente establecemos instancias y mandamos a llamar a los métodos

```
from RSA import RSA

rsa = RSA()

rsa.key_generation()

password = rsa.password_generator()

print(f"La contraseña es:{password}")

cipher_password = rsa.cipher("PublicKey.txt", password)

print(f'La contraseña cifrada es: {cipher_password}')

#Decifrado
decipher_password = rsa.decipher('PrivateKey.txt', ciphertext=cipher_password)

print(f'La contraseña decifrada es:{decipher_password}')

```

Prueba 1

```
D:\Programs\Anaconda\python.exe "D:\ESCOM\10°Semestre\Cripto 2\
La contraseña es:T(a|5*K
La contraseña cifrada es: CRhRQvEhoLKlC87BAVxaG5ve4f3z
La contraseña decifrada es:T(a|5*K

Process finished with exit code 0
```

Prueba 2

```
D:\Programs\Anaconda\python.exe "D:\ESCOM\10°Semestre\Cripto 2\
La contraseña es:r:9@7::
La contraseña cifrada es: Fr38+vQOy8dF7O1kyqXcuST/nXIo
La contraseña decifrada es:r:9@7::

Process finished with exit code 0
```

Prueba 3

```
D:\Programs\Anaconda\python.exe "D:\ESCOM\10°Semestre\Cripto 2\
La contraseña es:Pc/oNjR
La contraseña cifrada es: B+0y+1zZKUrM1m3PCEtcgóc76OHI
La contraseña decifrada es:Pc/oNjR

Process finished with exit code 0
```