RSA
Seminario de Solución de Problemas de Métodos Matemáticos I

Navarro Presas Moisés Alejandro - 215861509

May 17, 2016

Contents

1 RSA

RSA es un sistema criptográfico de clave publica, es decir que utiliza encriptación asimétrica. En la actualidad es el sistema de encriptación mas utilizado por su robustez; ya que para logar romperlo, con computadoras normales, el proceso puede llevar años

1.1 Algoritmo

Obtención de llaves

1. Se eligen dos números primos distintos p y q
2. Se calcula $n = pq$
3. Se calcula $\varphi(n) = (p-1)(q-1)$
4. Se elige un numero positivo e menor que $\varphi(n)$ y coprimo de $\varphi(n)$
5. Se determina d tal que $d \cdot e \equiv 1 \pmod{\varphi(n)}$

La llave publica $= (n, e)$
La llave privada $= (n, d)$

Cifrado
$c \equiv m^e \pmod{n}$

Descifrado
$c \equiv m^d \pmod{n}$

2 Programa

El programa está desarrollado en python, y a su vez está dividido en tres archivos:

1. Generación de llaves (*keygen.py*)
2. Encriptación (*cipher.py*)
3. Desencriptación (*decipher.py*)

2.0.1 keygen.py

Lo primero que debemos hacer es obtener una lista con n números primos, esto lo hacemos mediante la criba de Eratóstenes

```python
def get_primes(n):
    primes = list(range(2, n))
    for number in primes:
        for mult in list(range(number*number, n, number + number)):
            primes.remove(mult)
    return primes
```

Lo que se hace en este algoritmo es recorrer una lista de números (del 2 hasta el máximo deseado) e ir descartando todos aquellos que son múltiplos de un numero ya evaluado.

De esta lista de números primos se eligen p y q

```python
p = 0
while not p in primes:
    p = int(input("Ingrese un numero primo 'p'>"))

q = 0
while not q in primes or p == q:
    q = int(input("Ingrese un numero primo 'q'>"))
```

Posteriormente se calcula tanto n como $\varphi(n)$

```python
n = p * q

phi_n = (p - 1) * (q - 1)
```

Después se generan las posibles llaves publicas

```python
public_keys = [number for number in range(2, phi_n) if gcd(number, phi_n) == 1]
```

Esto se logra revisando todos los números entre 2 y $\varphi(n)$, tomando aquellos cuyo gcd con respecto de $\varphi(n)$ sea igual a 1

Se selecciona una de estas posible llaves publicas (e)

```python
e = 0
while not e in public_keys:
    e = int(input("Ingrese un numero coprimo de " + str(phi_n) + " 'e'>"))
```

Finalmente se calcula el inverso multiplicativo de $e \pmod{n}$

```python
def inverse(e, n):
    i = 1
    while (i * e) % n != 1:
        i += 1
    return i
```

2.0.2 cipher.py

El proceso de cifrado es bastante simple, tan solo hay que elevar el dato a cifrar (c) a la potencia e y posteriormente obtener su modulo n

```python
n = int(input("Ingrese 'n'>"))
e = int(input("Ingrese 'e'>"))
c = int(input("Ingrese el dato a cifrar>"))

c **= e
c %= n

print("El dato cifrado es: " + str(c))
```

2.0.3 decipher.py

El proceso de para descifrar es similar al de cifrado, tan solo hay que elevar el dato a cifrar (c) a la potencia d y posteriormente obtener su modulo n

```python
n = int(input("Ingrese 'n'>"))
d = int(input("Ingrese 'd'>"))
c = int(input("Ingrese el dato a descifrar>"))

c **= d
c %= n

print("El dato descifrado es: " + str(c))
```