UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FCFM
FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

NOMBRE: ROMAN ALEJANDRO GALLEGOS MARQUEZ

MATRICULA: 1735449 CARREA: LSTI

CRIPTOGRAFÍA

F.C.F.M.

PIA

Tabla de contenido

INTRODUCCION

Este documento te ayudara a comprender mejor el funcionamiento del algoritmo de cifrado RSA descubriendo su origen, las matemáticas que existen detrás de él y mediante un programa realizado en Python te ayudara a poder visualizarlo en practica y si cuentas con los archivos de Python, podrás interactuar con ellos aprendiendo así de una manera mucho más divertida y dinámica.

RSA es un algoritmo de cifrado asimétrico cifrado en bloques, fue creado por 3 personas Ron **R**ives, Adi **S**hamir y Leonard **A**dleman, si son observadores pueden notar que el nombre surge debido a la primera letra del apellido de sus creadores.

Así damos comienzo al documento, no sin antes agradecer por la atención prestada, muchas gracias y espero que sea de su agrado.

RSA

¿QUE ES RSA?

Este es un algoritmo asimétrico cifrador de bloques, que utiliza una clave pública, la cual se distribuye (en forma autenticada preferentemente), y otra privada, la cual es guardada en secreto por su propietario.

Una clave es un número de gran tamaño, que una persona puede conceptualizar como un mensaje digital, como un archivo binario o como una cadena de bits o bytes.

Cuando se envía un mensaje, el emisor busca la clave pública de cifrado del receptor y una vez que dicho mensaje llega al receptor, éste se ocupa de descifrarlo usando su clave oculta.

UN POCO DE HISTORIA

Un poco de historia, este es inventado 1977 y lleva la primera letra del apellido de sus creadores Ron Rives, Adi Shamir y Leonard Adleman, formando asi el algoritmo RSA, esta basa su fortaleza en la dificultad de factorizar de dos numeros primos grandes, algo que es de alta dificulta para la capacidad de las computadoras.

LAS MATEMATICAS DETRÁS DE RSA

Conforme pasaba el tiempo se fue convirtiendo en un estandar y funciona de la siguiente manera:

Alice → n = p*q

Bob → p*q

Siendo p y q, números primos de 512 bits aun que en la actualidad es recomendable que sea de 1024 bits, los valores de p y q seran las claves secretas.

La idea es complicada pero una forma sencilla de explicarlo es:

Escoger dos primos (p y q)

Calcular $n = p \cdot q$

Calcular $\varphi(n) = (p-1)(q-1)$

Calcular e tal que sea menor a φ(n) y no tengan factores primos en común.

Calcular d para que el resto de la división *de* entre *φ(n)* sea 1

Clave pública: (e, n)

Clave privada: (d, n)

Sean p, q primos

$n = p \cdot q$

$\varphi(n) = (p-1)(q-1)$

e tal que:

$< e < \varphi(n) \wedge mcd(e, \varphi(n)) = 1$

d tal que:

$de \equiv 1 (mod\, \varphi(n))$

Y Si tenemos:

p = 3, q = 11

N = 3 x 11 = 33

φ(n) = (3-1)(11-1) = 20

e = 7 (menor a φ(n) y coprimo con él)

$7d \equiv 1\ (mod\, 20) \rightarrow d = 4$

Clave pública: **(7, 33)**

Clave privada: **(4, 33)**

Tenemos dos primos p y q, p=3 y q = 11, los multiplicaremos entre ellos, dandonos N= p*q = 3 * 11 = 33, claro que si fueramos el atacante lo resolveriamos muy rapido con una claves asi, ya que dos numeros que multiplicados entre si me den

33, bueno esto si es sencillo para este paso pero recordemos que p y q son numeros primos enormes.

Bueno ahora usaremos la formula de

$$\varphi(n) = (p-1)(q-1)$$

Dandonos:

$\varphi(n) = (3-1)(11-1) = 20$

e = 7 (menor a $\varphi(n)$ y coprimo con él)

Ademas debemos de escoger un ‘e’ que sea menor a phi ϕ y ahora hay que calcular D

$$7d \equiv 1 \,(mod\, 20) \rightarrow d = 4$$

Clave pública: **(7, 33)**

Clave privada: **(4, 33)**

7 por d es congruente a 1 mod 20 y el resultado nos da 4, ahora tendremos nuestra clave publica y privada (7,33) y (4, 33), y ahora le puedo dar la clave publica a las personas que vayan a recibir un mensaje,y ellos apartir del (7,33) no podran sacar que yo tengo el (4,33).

Ahora para encriptar y desencriptar es de la siguiente manera:

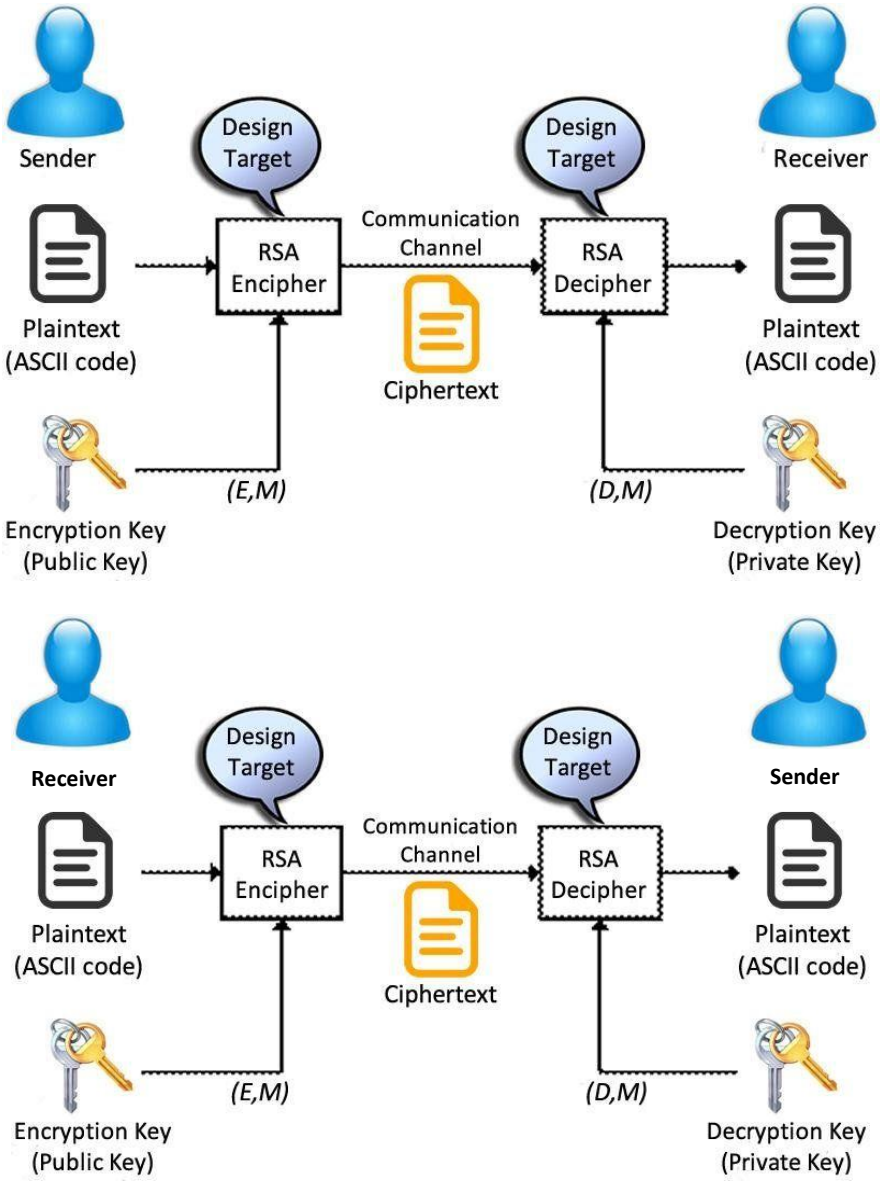
Encriptar: $\mathrm{c}=\mathrm{m}^e \,(mod\, n)$

Desencriptar: $m = c^d \,(mod\, n)$

Parece sencillo pero recordemos que ‘e’ puedes ser un numero mucho mayor y para descifrar seria con el mensaje encriptado elevado a D mod n.

SEGURIDAD RSA

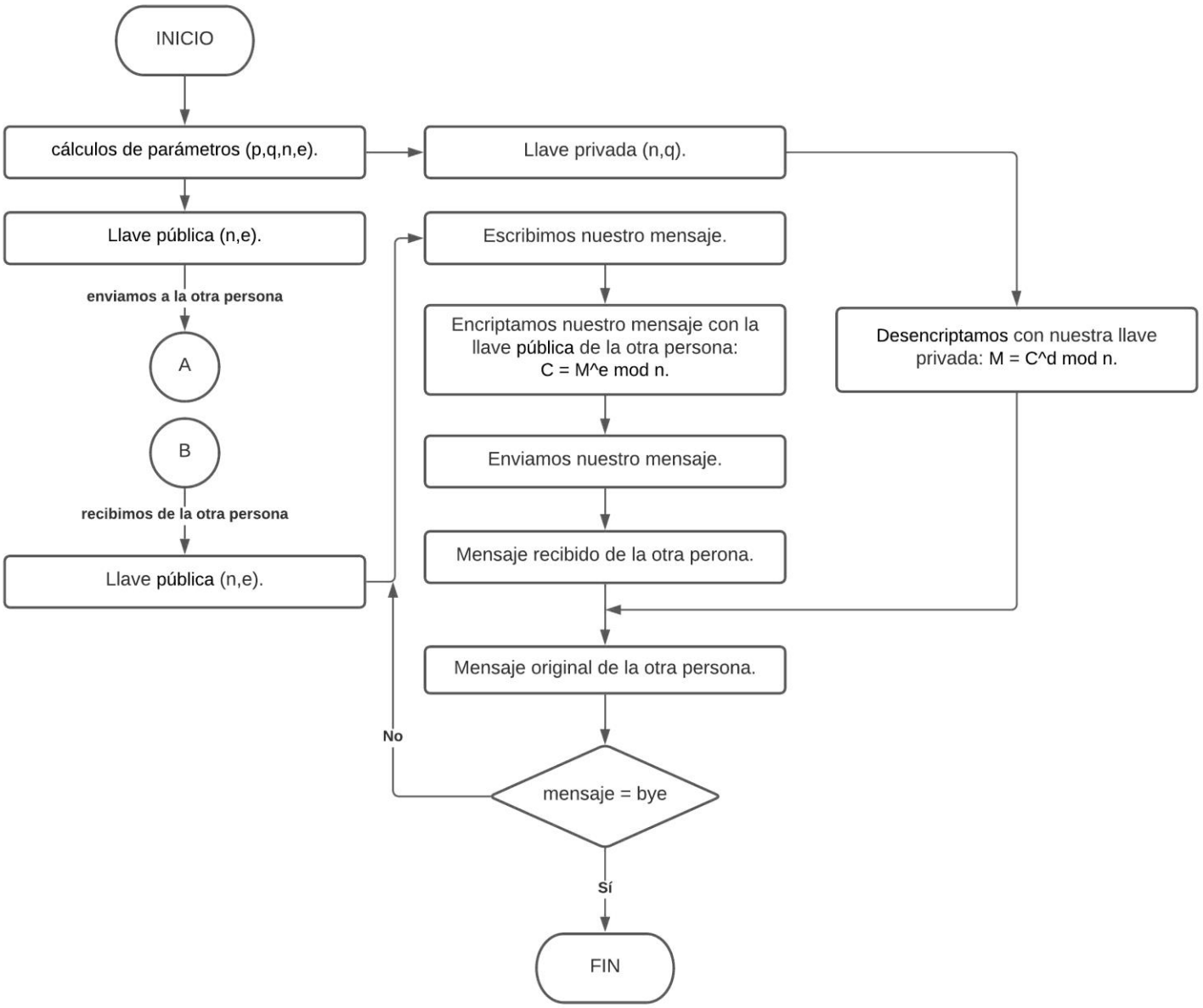
La seguridad de este algoritmo radica en que no hay maneras rápidas conocidas de factorizar un número grande en sus factores primos utilizando computadoras tradicionales.



ALGORITMO DEL PROGRAMA

1. INICIO
2. Hacemos los cálculos de parámetros (p,q,n,e)
3. Sacamos la Llave pública (n,e).
4. Sacamos la Llave privada (n,q).
5. Mandamos nuestra llave.
6. Recibimos la llave publica de la otra persona.
7. Escribimos nuestro mensaje
8. Ciframos nuestro mensaje con la llave publica de la otra persona:
   C → M^e mod n.
9. Enviamos nuestro mensaje.
10. Recibimos el mensaje de la otra persona.
11. Desencriptamos con nuestra llave privada:
    M → C^d mod n.
12. Mensaje original de la otra persona.
13. Si mensaje original de la otra persona → bye
    Si no volver a paso 7.
14. FIN

DIAGRAMA DE FLUJO DEL PROGRAMA

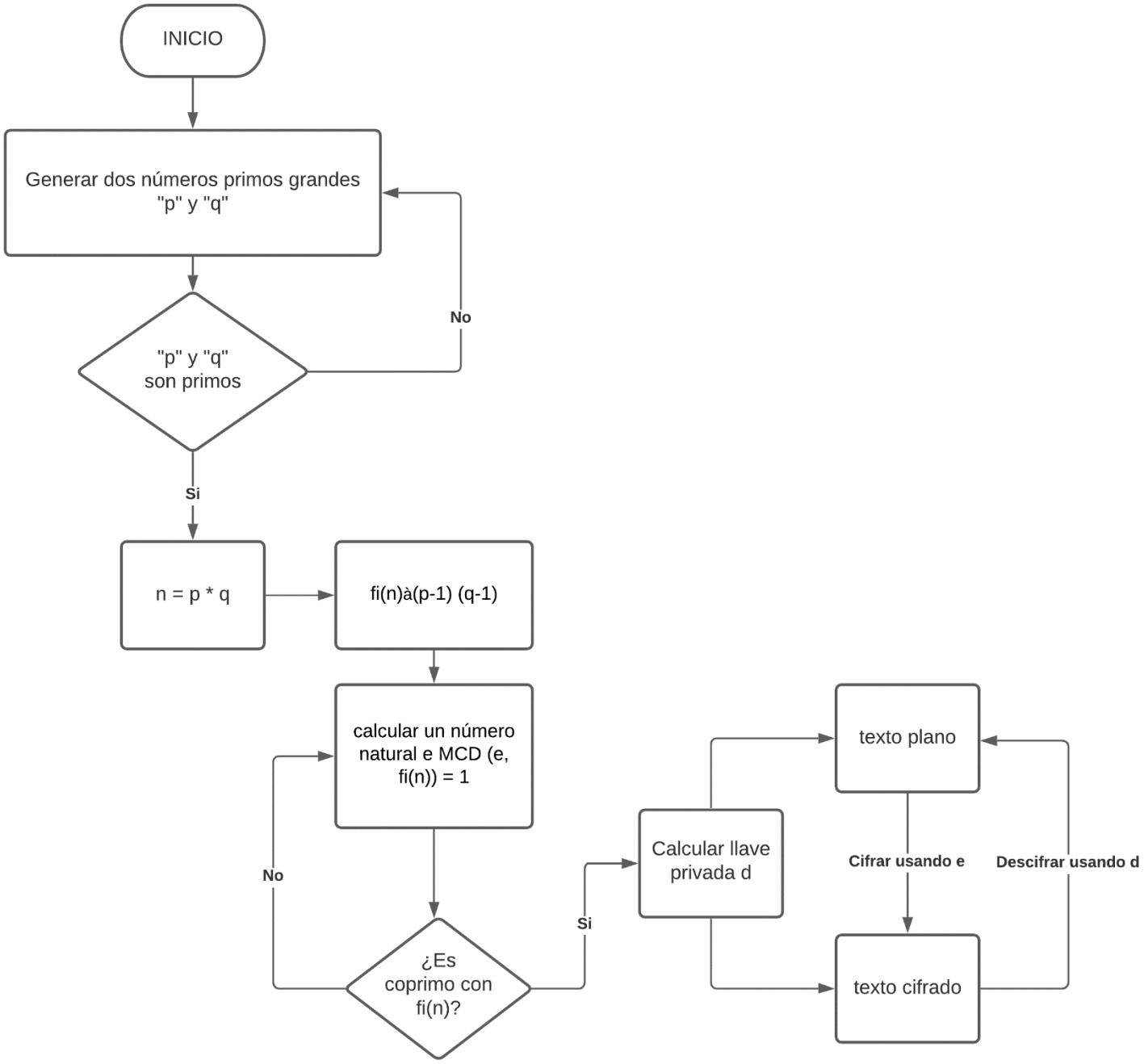


ALGORITMO RSA

15. Inicialmente es necesario generar aleatoriamente dos **números primos** grandes o elegirlos tambien, a los que llamaremos p y q.
16. A continuación, calcularemos n como producto de p y q: n → p * q.
17. Se calcula SI: fi(n)→(p-1) (q-1).
18. Se calcula un número natural e de manera que MCD (e, fi(n)) = 1, es decir e debe ser primo relativo de fi(n).
    Es lo mismo que buscar un numero impar por el que dividir fi(n) que de cero como resto.
19. Mediante el algoritmo extendido de Euclides se calcula d: e.d mod fi(n)=1
    Puede calcularse d=((Y*fi(n)) +1) /e para Y=1, 2, 3, . . ., d hasta encontrar un d entero.
20. El par de números (e,n) son la clave pública.
21. El par de números (d,n) son la clave privada.
22. Cifrado: La función de cifrado es C = M^e mod n.
23. Descifrado: La función de descifrado es M = C^d mod n .

Esto solo es el bloque de RSA.

DIAGRAMA DE FLUJO RSA



PROGRAMA PYTHON

REQUISITOS

Debemos de tener instalada la última versión de Python en nuestros equipos donde vayamos a correr los programas.

MODO DE USO

WINDOWS

Abriremos dos ventanas de cmd / terminal y nos ubicaremos en la carpeta donde se encuentren nuestros archivos, lo vamos a correr de forma local, en una pantalla de cmd / terminal donde vamos a correr “PERSONA1-SERVIDOR.py” para esto debemos poner: “python persona1-servidor.py” y darle ENTER, recordando que debemos tener python además de estar en la carpeta donde están nuestros archivos .py, ahora haremos lo mismo para “PERSONA2-CLIENTE.py” solo que en esta ventana de cmd / terminal pondremos “python persona2-cliente.py”.

LINUX

Abriremos dos de nuestras terminales de Linux, debemos tener nuestra última versión de python y estar ubicados dentro de la carpeta donde tenemos los archivos .py, ahora en una pondremos “python3 persona1-servidor.py” y dar ENTER, lo mismo para nuestra segunda terminal, pero ahora con el segundo archivo “python3 persona2-cliente.py”

MAC

Abriremos una terminal de mac, para ello debes irte a “APLICACIONES”, después en “UTILIDADES” y por ultimo “TERMINAL”, ya vienen con un intérprete de python pero es importante tenerlo actualizado, ahora lo que haremos es poner “python persona1-servidor.py” y dar ENTER, lo mismo para nuestra segunda terminal, pero ahora con el segundo archivo “python persona2-cliente.py”

“ES IMPORTANTE PRIMERO CORRER EL ARCHIVO .PY LLAMADO PERSONA1-SERVIDOR PARA TODAS LAS OPCIONES.”

PROGRAMA

Corremos primero el servidor y después el cliente:

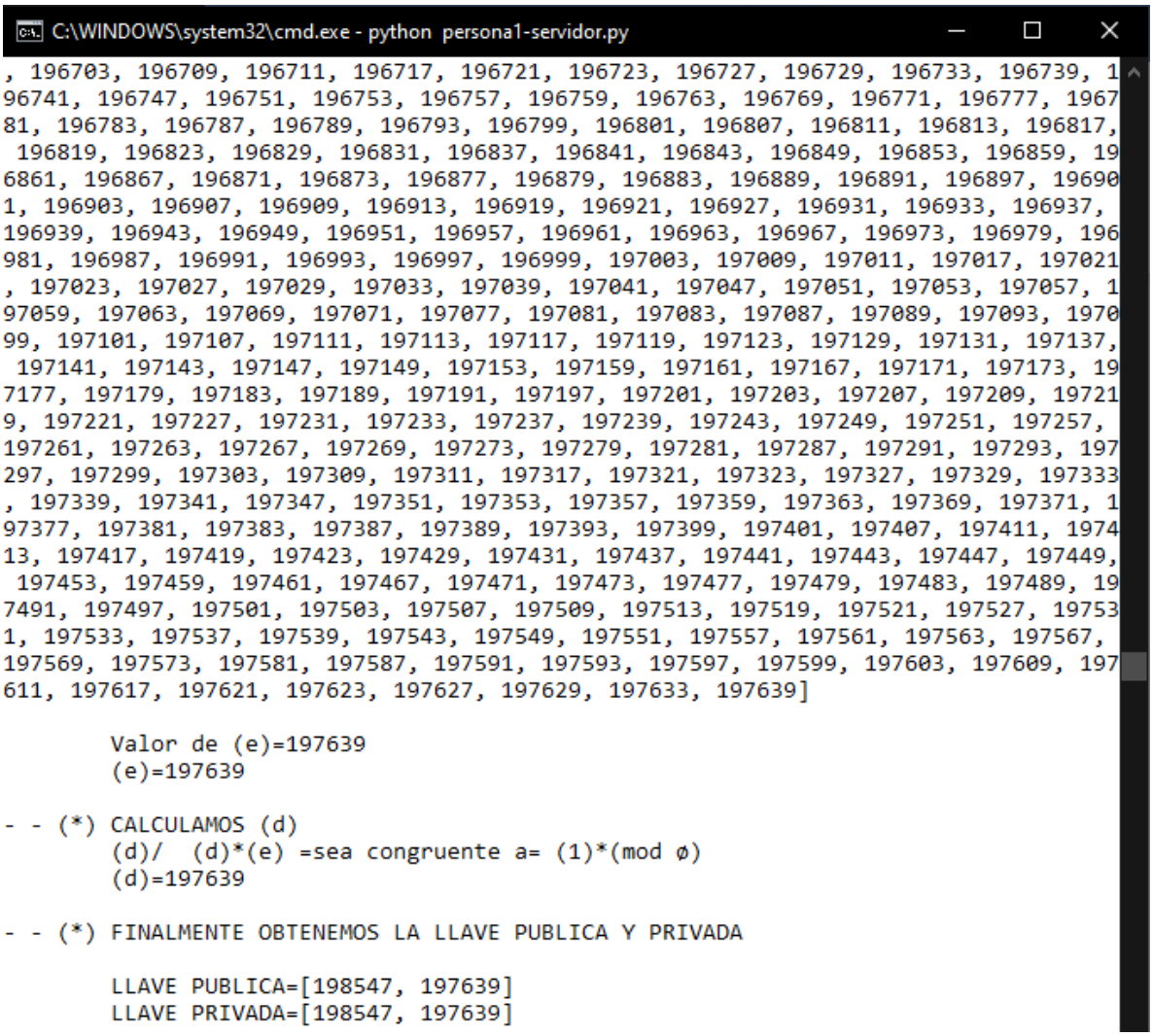
Para cliente ingresamos valores, debemos usar números primos grandes para hacer la operación y garantizar que con una computadora normal no pueda descifrarlo, ya que como sabemos los números primos son complejos de procesar.

Para servidor:


```
- - (*) ELEGIMOS VALORES DE NUMEROS PRIMOS PARA (p) y (q)

LISTA DE NUMEROS PRIMOS =[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 5
3, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139,
 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 23
3, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331,
337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433
, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541]

        Valor de (p)=367
        Valor de (q)=541
        (p)=367
        (q)=541

- - (*) CALCULAMOS EL VALOR DE (n)
        (n)=(p)*(q)
        (n)=(367)*(541)
        (n)=198547

- - (*) CALCULAMOS (ø)
        (ø)=(p-1)*(q-1)
        (ø)=(367-1)*(541-1)
        (ø)=197640

- - (*) CALCULAMOS (e)
        (e)/  1<e<ø and mcd(e,ø)==1
```

P = 367 y Q = 541

No dará muchos valores que son enormes y escogeremos uno para e:



Ahora esperaremos la llave publica de la otra persona (cliente) – TITA:

CLIENTE - TITA:

P = 199 y Q = 211

No dará muchos valores que son enormes y escogeremos uno para e:

```
40837, 40841, 40847, 40849, 40853, 40861, 40867, 40871, 40877, 40879, 40883, 40889,
 40891, 40897, 40903, 40907, 40913, 40919, 40921, 40927, 40933, 40937, 40939, 40949
, 40951, 40961, 40963, 40967, 40969, 40973, 40979, 40981, 40987, 40991, 40993, 4100
3, 41009, 41011, 41017, 41021, 41023, 41029, 41033, 41039, 41047, 41051, 41053, 410
57, 41059, 41071, 41077, 41081, 41087, 41089, 41093, 41099, 41101, 41113, 41117, 41
119, 41123, 41131, 41137, 41141, 41143, 41147, 41149, 41159, 41161, 41171, 41177, 4
1179, 41183, 41189, 41191, 41197, 41201, 41203, 41207, 41213, 41219, 41221, 41227,
41231, 41233, 41243, 41249, 41257, 41263, 41267, 41269, 41273, 41281, 41287, 41291,
 41297, 41299, 41303, 41309, 41311, 41317, 41323, 41329, 41333, 41339, 41341, 41347
, 41351, 41353, 41357, 41359, 41369, 41381, 41383, 41387, 41389, 41399, 41401, 4140
7, 41411, 41413, 41417, 41423, 41429, 41431, 41441, 41443, 41449, 41453, 41467, 414
71, 41473, 41477, 41479, 41483, 41491, 41497, 41501, 41507, 41509, 41513, 41519, 41
521, 41527, 41533, 41537, 41539, 41543, 41549, 41551, 41557, 41561, 41563, 41567, 4
1579]

        Valor de (e)=41563
        (e)=41563

- - (*) CALCULAMOS (d)
        (d)/  (d)*(e) =sea congruente a= (1)*(mod ø)
        (d)=19567

- - (*) FINALMENTE OBTENEMOS LA LLAVE PUBLICA Y PRIVADA

        LLAVE PUBLICA=[41989, 41563]
        LLAVE PRIVADA=[41989, 19567]
 ----------------------------------------
  * * * * * * * * * * * * * * * * * * *
 ----------------------------------------
                      .--.
                     |o_o |
                     |:_/ |
                    //   \ \
                   (|     | )
                  /'\_   _/`\
                  \___)=(___/

(-)TITA <3 :
```

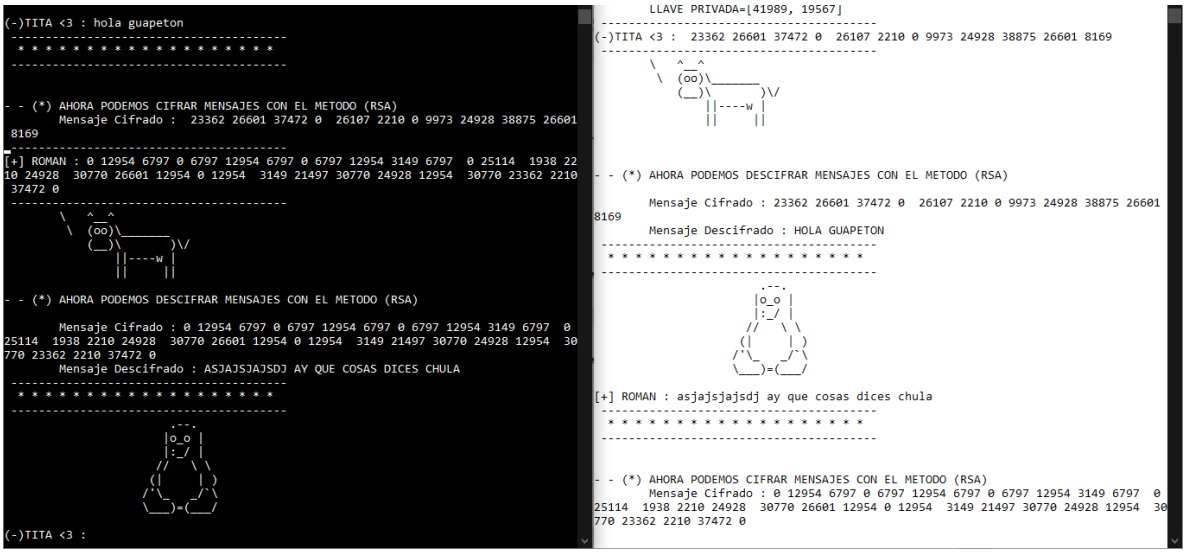
Ya se intercambiaron la llave y ahora podemos enviar mensaje desde TITA:

MENSAJES:

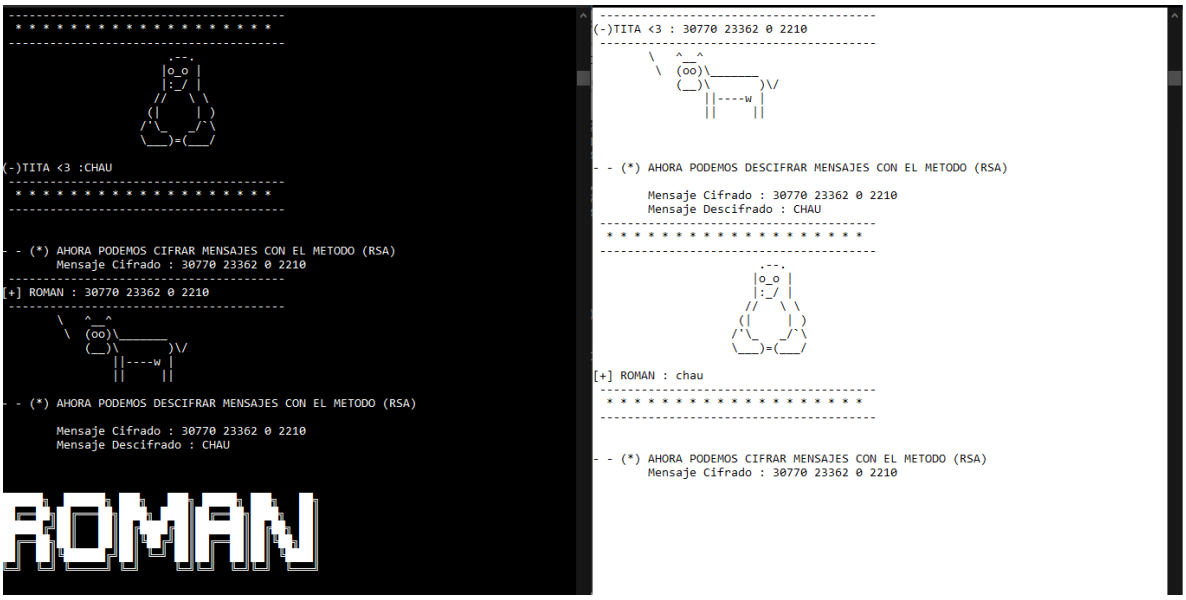
```
 --------------------------------------
  * * * * * * * * * * * * * * * * * * *
 --------------------------------------
                      .---.
                     |o_o |
                     |:_/ |
                    //   \ \
                   (|     | )
                  /'\_   _/`\
                  \___)=(___/

(-)TITA <3 : hola guapeton
 --------------------------------------
  * * * * * * * * * * * * * * * * * * *
 --------------------------------------


- - (*) AHORA PODEMOS CIFRAR MENSAJES CON EL METODO (RSA)
        Mensaje Cifrado :  23362 26601 37472 0  26107 2210 0 9973 24928 38875 26601
 8169
```

Enviamos y recibimos mensajes.

```
 ----------------------------------------
(-)TITA <3 :  23362 26601 37472 0  26107 2210 0 9973 24928 38875 26601 8169
 ----------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/
                ||----w |
                ||     ||



- - (*) AHORA PODEMOS DESCIFRAR MENSAJES CON EL METODO (RSA)

        Mensaje Cifrado : 23362 26601 37472 0  26107 2210 0 9973 24928 38875 26601
8169
        Mensaje Descifrado : HOLA GUAPETON
 ----------------------------------------
  * * * * * * * * * * * * * * * * * * * *
 ----------------------------------------
                       .---.
                      |o_o |
                      |:_/ |
                     //   \ \
                    (|     | )
                   /'\_   _/`\
                   \___)=(___/

[+] ROMAN :
```



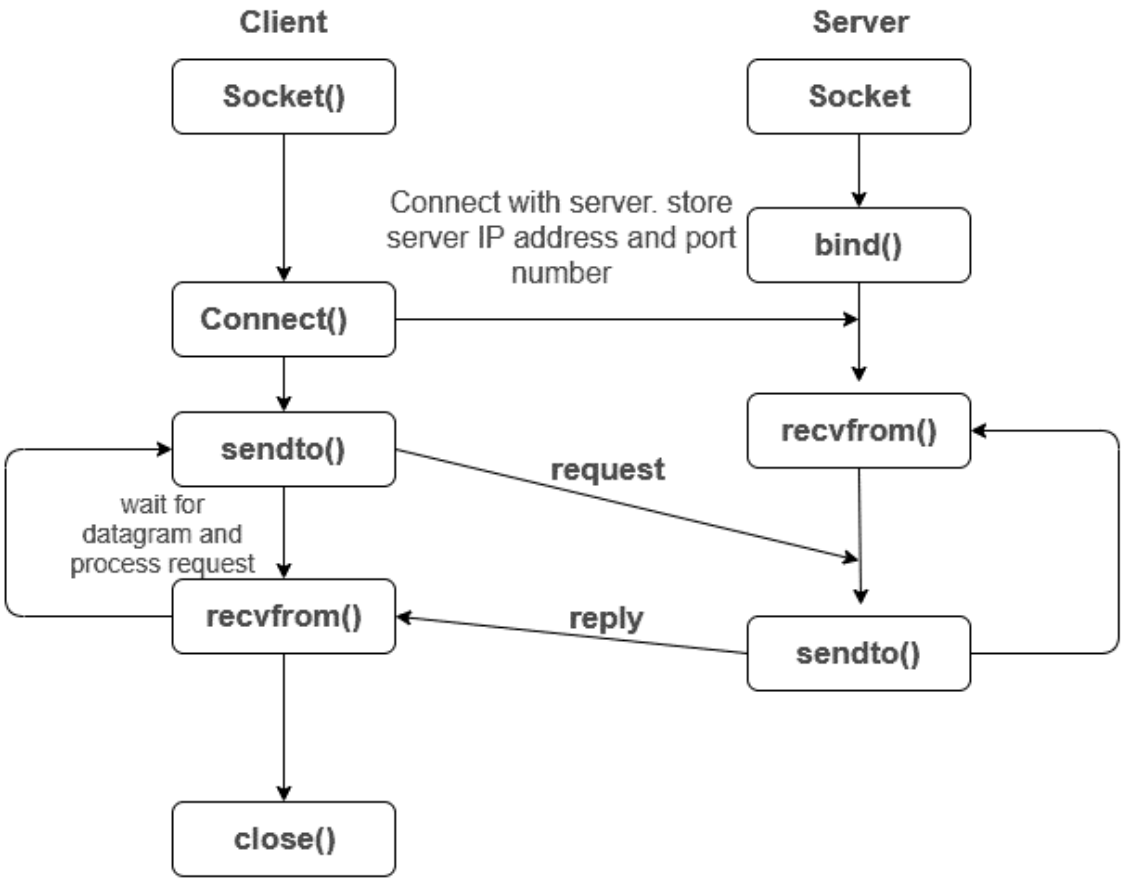
Y así pueden estar mandándose mensajes hasta recibir un bye, chau o adios:



Cerrando así el programa.

SOCKET

El funcionamiento de socket es sencillo y lo usamos para establecer la conexión.



PICKLE

Ya que el socket transporta bytes, tienes que convertir la lista a una secuencia de bytes. Típicamente hay dos formas para ello:

Usar un formato binario conocido por python. Para esto suele usarse el módulo pickle que es capaz de convertir cualquier tipo de datos pyhon a una serie de bytes, normalmente con el fin de guardarlo en disco, pero también, por qué no, para transmitirlo por un socket. Esto permitiría a quien lo recibe "recuperar" la variable enviada por si la quiere procesar localmente de alguna forma.

METODO RSA:

```
def pyq():

    p=int(input("\tValor de (p)="))
    while verifica_primo(p)==False:
        print("\t(p) tiene que ser un numero primo !!!")
        p=int(input("\tValor de (p)="))
    q=int(input("\tValor de (q)="))
    while verifica_primo(q)==False or q==p:
        print("\t(q) tiene que ser un numero primo diferente de (p) !!!")
        q=int(input("\tValor de (q)="))
    lpq=[p,q]
    return lpq
```

Con esta función me encargo de que los valores de p y q sean primos.

```
def calculae(ø):
    e=2
    le=[]
    while e>1 and e<ø :
        if mcd(e,ø)==1:
            le.append(e)
            e=e+1
        else:
            e=e+1
    print("\nVALORES PARA (e)="+str(le))
    e=int(input("\n\tValor de (e)="))
    while mcd(e,ø)!=1:
        print("\n\tEliga un valor de la lista !!!")
        e=int(input("\n\tValor de (e)="))
    return e
```

Con esta función calcularemos el valor de fi.

```
def mcd(e,ø):
    m=ø%e
    while m!=0:
        ø=e
        e=m
        m=ø%e
    return e
```

Con esta función encontraremos el mínimo común divisor con e y fi.

```
def congruente(e,ø):
    k=1
    m=(1+(k)*(ø))%(e)
    while m!=0:
        k=k+1
        m=(1+(k)*(ø))%(e)
    d=int((1+(k)*(ø))/(e))
    return d
```

Revisar si es congruente.

CIFRAR Y DESCIFRAR:

```
def cifrarmensaje(msj,key):
    msj=msj.upper()
    lm=msj.split(" ")
    cmc=""
    lmc=[]
    for i in lm:
        pal=cifrarpalabra(i,key)
        lmc.append(pal)
    for j in lmc:
        cmc=cmc+str(j)+" "
    return cmc

def cifrarpalabra(m,k):
    lpc=[]
    lp=[]
    n,e=k
    cpc=""
    for i in m:
        x=buscarpos(i)
        lp.append(x)
    for j in lp:
        c=(j**e)%n
        lpc.append(c)
    for k in lpc:
        cpc=cpc+str(k)+" "
    return cpc
```

Ciframos el mensaje y ciframos las palabras, esta función la utilizaremos más abajo con los parámetros del mensaje y key_public de la otra persona.

```
    while True:
        print(" ----------------------------------------"+"\n  * * * * * * * * * * * *
\t\t        .--.
\t\t       |o_o |
\t\t       |:_/ |
\t\t      //   \ \\
\t\t     (|     | )
\t\t    /'\_   _/`\\
\t\t    \___)=(___/
""")
        pregunta = input("(-)TITA <3 : ")
        print(" ----------------------------------------"+"\n  * * * * * * * * * * * *

        # MANDA Y CONTESTA PERO PRIMERO MANDA
        print("\n- - (*) AHORA PODEMOS CIFRAR MENSAJES CON EL METODO (RSA)")
        #mensaje=input("\n\tMensaje : ")
        mensaje_cifrado = cifrarmensaje(pregunta,key_public_roman)
        print("\tMensaje Cifrado : "+str(mensaje_cifrado))
        TCPCliente.sendto(str.encode(mensaje_cifrado), servidor)
```

```
def buscarpos(x):
    alf="ABCDEFGHIJKLMNÑOPQRSTUVWXYZ"
    c=0
    for i in alf:
        if x==i:
            return c
        else:
            c=c+1
```

Buscamos los campos correspondientes al alfabeto importante considerar la Ñ.

```
def descifrarmensaje(msj,key):
    msj=msj.upper()
    lm=msj.split("  ")
    cmc=""
    lmc=[]
    for i in lm:
        pal=descifrarnumero(i,key)
        lmc.append(pal)
    for j in lmc:
        cmc=cmc+str(j)+" "
    return cmc
```

Desciframos el mensaje con la key privada de nosotros que la consideramos más abajo:

```
 print(" ----------------------------------------")
 print("(-)TITA <3 : ", mensaje.decode())
 print(""" ----------------------------------------
     \   ^__^
      \  (oo)\_______
         (__)\       )\/\
             ||----w |
             ||     ||

)
 ## RECIBE
 print("\n- - (*) AHORA PODEMOS DESCIFRAR MENSAJES CON EL METODO (RSA)")
 mensaje_cifrado=input("\n\tMensaje Cifrado : ")
 mensaje_descifrado = descifrarmensaje(mensaje_cifrado,key_private)
 print("\tMensaje Descifrado : "+str(mensaje_descifrado))
```

GITHUB:

https://github.com/RomanAlejandro/RSA-python-chat

FUENTES:

https://docs.python.org/es/3.9/library/socket.html

https://docs.python.org/es/3/library/pickle.html

https://rico-schmidt.name/pymotw-3/socket/tcp.html

https://pythontic.com/modules/socket/udp-client-server-example

https://stackoverflow.com/questions/53576851/socket-programming-in-python-using-pickle

https://www.youtube.com/watch?v=Lbfe3-v7yE0&t=324s&ab_channel=sentdex

https://docs.python.org/3/library/stdtypes.html

https://www.youtube.com/watch?v=ojX9YPxhoX0&list=LL&index=2&t=166s&ab_channel=LuisMunoz

https://www.youtube.com/watch?v=kiowXySiuP8&list=LL&index=29&ab_channel=DavidAlejandroNinaRojas

HERRAMIENTA DE ELABORACIÓN DE DIAGRAMAS DE FLUJO:

https://lucid.app