

NOMBRE: ROMAN ALEJANDRO GALLEGOS MARQUEZ

MATRICULA: 1735449 CARREA: LSTI

**CRIPTOGRAFÍA**

F.C.F.M.

PIA

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza baja

Tabla de contenido

[**INTRODUCCION** 3](#_Toc87892649)

[**RSA** 4](#_Toc87892650)

[**¿QUE ES RSA?** 4](#_Toc87892651)

[**UN POCO DE HISTORIA** 4](#_Toc87892652)

[**LAS MATEMATICAS DETRÁS DE RSA** 5](#_Toc87892653)

[**SEGURIDAD RSA** 7](#_Toc87892654)

[**ALGORITMO DEL PROGRAMA** 8](#_Toc87892655)

[**DIAGRAMA DE FLUJO DEL PROGRAMA** 9](#_Toc87892656)

[**ALGORITMO RSA** 10](#_Toc87892657)

[**DIAGRAMA DE FLUJO RSA** 11](#_Toc87892658)

[**PROGRAMA PYTHON** 12](#_Toc87892659)

[**REQUISITOS** 12](#_Toc87892660)

[**MODO DE USO** 12](#_Toc87892661)

[WINDOWS 12](#_Toc87892662)

[LINUX 12](#_Toc87892663)

[MAC 12](#_Toc87892664)

[PROGRAMA 13](#_Toc87892665)

[**SOCKET** 20](#_Toc87892666)

[**PICKLE** 20](#_Toc87892667)

[METODO RSA 21](#_Toc87892668)

[CIFRAR Y DESCIFRAR 22](#_Toc87892669)

[FUENTES 25](#_Toc87892670)

[HERRAMIENTA DE ELABORACIÓN DE DIAGRAMAS DE FLUJO 26](#_Toc87892671)

# **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:

Texto

Descripción generada automáticamenteTexto, Carta

Descripción generada automáticamente

Y Si tenemos:

Texto, Carta

Descripción generada automáticamente

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 Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Dandonos:

Texto

Descripción generada automáticamente con confianza media

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

Texto

Descripción generada automáticamente

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:

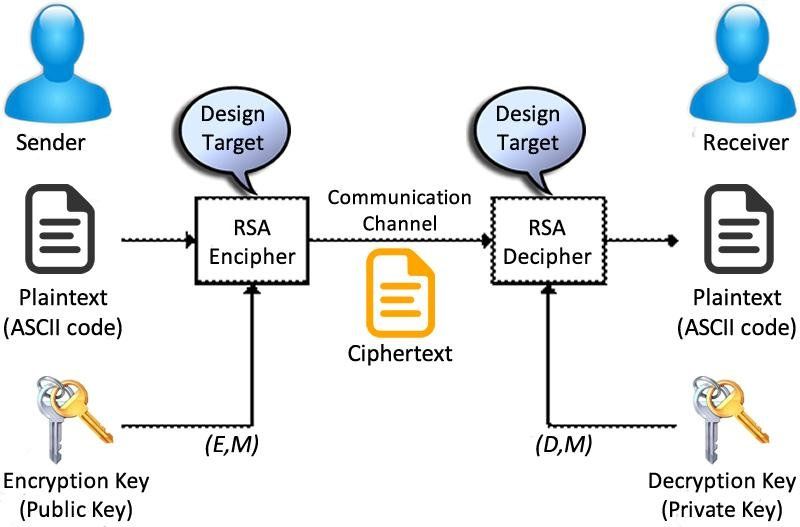
Texto

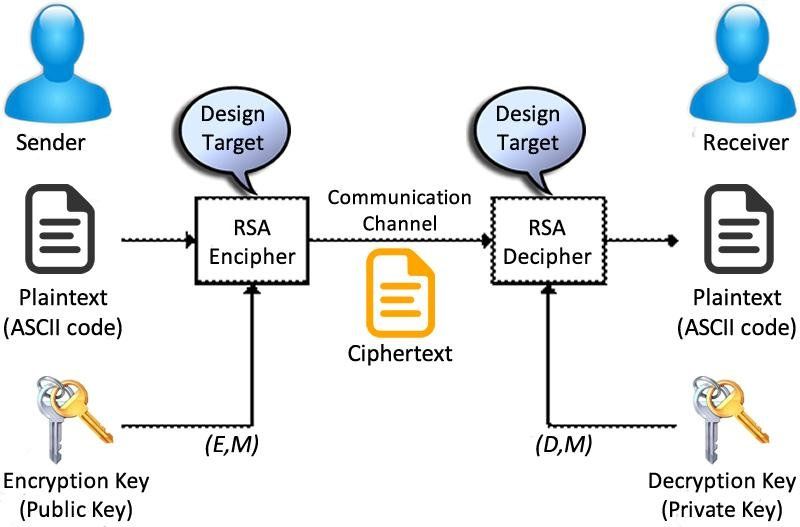
Descripción generada automáticamente con confianza media

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.





**Sender**

**Receiver**

# **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.

1. Enviamos nuestro mensaje.
2. Recibimos el mensaje de la otra persona.
3. Desencriptamos con nuestra llave privada:

M 🡪 C^d mod n.

1. Mensaje original de la otra persona.
2. Si mensaje original de la otra persona 🡪 bye

Si no volver a paso 7.

1. FIN

# **Diagrama Descripción generada automáticamenteDIAGRAMA DE FLUJO DEL PROGRAMA**

# **ALGORITMO RSA**

1. Inicialmente es necesario generar aleatoriamente dos **números** **primos** grandes o elegirlos tambien, a los que llamaremos p y q.
2. A continuación, calcularemos n como producto de p y q: n 🡪 p \* q.
3. Se calcula SI: fi(n)🡪(p-1) (q-1).
4. 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.

1. 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.
2. El par de números (e,n) son la clave pública.
3. El par de números (d,n) son la clave privada.
4. Cifrado: La función de cifrado es C = M^e mod n.
5. 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:

Interfaz de usuario gráfica, Código QR

Descripción generada automáticamente

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:

Texto

Descripción generada automáticamente

P = 367 y Q = 541

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

Texto

Descripción generada automáticamente

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

CLIENTE - TITA:

Texto

Descripción generada automáticamente

P = 199 y Q = 211

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

Texto

Descripción generada automáticamente

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

MENSAJES:

Imagen que contiene Texto

Descripción generada automáticamente

Enviamos y recibimos mensajes.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

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

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Cerrando así el programa.

## **SOCKET**

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

Diagrama

Descripción generada automáticamente

## **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:

Texto

Descripción generada automáticamente

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

Texto

Descripción generada automáticamente

Con esta función calcularemos el valor de fi.

Tabla

Descripción generada automáticamente con confianza baja

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

Texto, Carta

Descripción generada automáticamente

Revisar si es congruente.

CIFRAR Y DESCIFRAR:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

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.

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Buscamos los campos correspondientes al alfabeto importante considerar la Ñ.

Texto

Descripción generada automáticamente

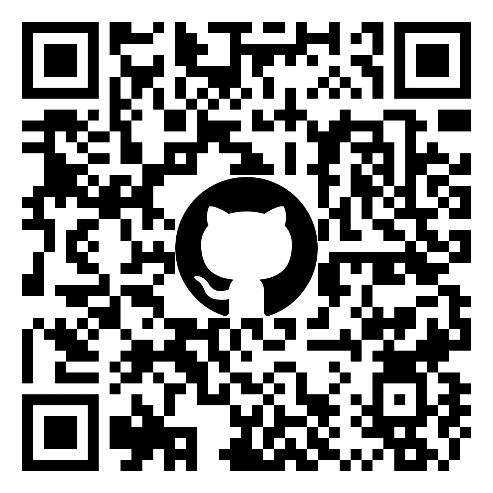
Desciframos el mensaje con la key privada de nosotros que la consideramos más abajo:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

 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>