

# Guia de Usuario

Proyecto Criptografia

March 2025

## Descripcion de la simulación

La simulación es un programa que permite a los jugadores realizar cálculos de forma segura y distribuida. Utiliza el esquema de Shamir para compartir secretos y la interpolación de Lagrange para reconstruir el resultado final. Los jugadores pueden realizar cálculos sin revelar sus valores secretos.

Donde lee un archivo por consola y cada simulación esta separada por un renglon en el archivo, donde cada renglon contiene los numeros a multiplicar separados por espacio. Cada renglon es una simulación diferente con un  $t$  (grado del polinomio) diferente.

## Instrucciones para ejecutar la simulación

Este documento proporciona instrucciones detalladas para ejecutar el programa desde la terminal en diferentes sistemas operativos.

### 1 Abrir la Terminal

#### 1.1 Windows

Para abrir la terminal en Windows, siga estos pasos:

- Presione **Win + R**, escriba `cmd` y presione **Enter**.
- También puede abrir el **PowerShell** buscando "PowerShell" en el menú de inicio.

#### 1.2 macOS

Para abrir la terminal en macOS:

- Presione **Cmd + Espacio** y escriba "Terminal".
- Seleccione la aplicación "Terminal" y ábrala.

### 1.3 Linux

Para abrir la terminal en Linux:

- Use el atajo **Ctrl + Alt + T**.
- O búsquela en el menú de aplicaciones.

## 2 Ejecutar el Programa

Primero, asegúrese de estar en la carpeta donde se encuentra el proyecto. Navegue hasta el directorio usando el comando `cd`:

```
cd ruta/del/proyecto
```

Luego, ejecute el siguiente comando:

```
python3 main.py -f "archivo.txt"
```

Reemplazando `archivo.txt` con el archivo de datos que desea usar el cual contiene los numero a multiplicar en filas, separados por espacio.

Luego, el mismo programá lo irá guiando por la terminal para seguir los pasos correctos.

## 3 Notas Adicionales

- Asegúrese de tener instalado **Python 3**. Puede verificarlo con:

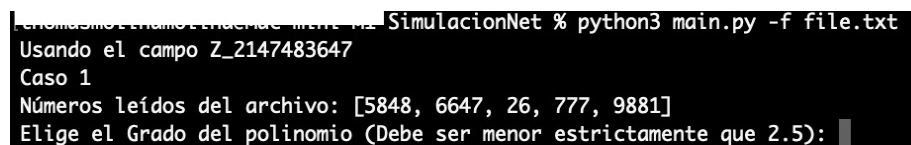
```
python3 --version
```

- Si el comando `python3` no funciona en Windows, intente con `python`:

```
python main.py -f "nombre-del-archivo.txt"
```

## 4 Explicacion de la ejecución

Una vez que haya seguido los pasos anteriores, el programa se ejecutará y le pedirá que ingrese el grado del polinomio. Este es diferente para cada simulación.



```
SimulacionNet % python3 main.py -f file.txt
Usando el campo Z_2147483647
Caso 1
Números leídos del archivo: [5848, 6647, 26, 777, 9881]
Elige el Grado del polinomio (Debe ser menor estrictamente que 2.5): █
```

Figure 1: El  $t$  es fijo para toda  $n$  simulación.

despues de colocar el grado del polinomio, el programa calculará todas las multiplicaciones y mostrará los resultados en la terminal.

Para motivos didacticos, el programa mostrará los primeros fragmentos y Luego la primera reparticion de los fragmentos para despues mostrar el resultado modulo el campo.

despues de esto volverla a preguntar sobre el valor del nuevo t en la nueva simulación.

```
Usando el campo Z_2147483647
Simulacion 1
Números leídos del archivo: [5848, 6647, 26, 777, 9881]
Elige el Grado del polinomio (Debe ser menor estrictamente que 2.5): 2
Configuración exitosa: 5 jugadores con polinomio de grado 2
Shares generados por el jugador 1:
[1148316827, 1203051793, 164210746, 179277333, 1248251554]

Shares generados por el jugador 2:
[1268989171, 2089938396, 315370675, 240253302, 1864586277]

Shares generados por el jugador 3:
[1213135853, 2030586931, 304869613, 330951193, 2108831671]

Shares generados por el jugador 4:
[1021429909, 1586841269, 1696234857, 1349610673, 546968717]

Shares generados por el jugador 5:
[1249424712, 692373022, 476338458, 601321020, 1067320708]

Fragmentos después de la repartición:
Party 1 con shares: [1148316827, 1268989171, 1213135853, 1021429909, 1249424712]
Party 2 con shares: [1203051793, 2089938396, 2030586931, 1586841269, 692373022]
Party 3 con shares: [164210746, 315370675, 304869613, 1696234857, 476338458]
Party 4 con shares: [179277333, 240253302, 330951193, 1349610673, 601321020]
Party 5 con shares: [1248251554, 1864586277, 2108831671, 546968717, 1067320708]
El resultado es: 1276999381 (mod 2147483647)
Simulacion 2
Números leídos del archivo: [3243, 243, 434]
Elige el Grado del polinomio (Debe ser menor estrictamente que 1.5): █
```

Figure 2: Cada party es un jugador que esta de forma ordenada.

## Descripcion de la red

La ejecución se puede correr de varias maneras, una de ellas es utilizando el archivo `main.py`. Para esto se debe estar dentro del archivo `Network` y correr el `main.py`. De esta manera el protocolo correra en la terminal solo para el usuario que corra el comando y debera conectarse manualmente con los otros jugadores que También ejecuten el archivo individualmente. Asi podremos correr el protocolo con diferentes dispositivos electronicos.

La otra opción de correr el archivo `.bash` (macOS) o `.cmd` (Windows) depende del sistema operativo que se está utilizando. Este archivo corre el protocolo en la terminal y se conecta automáticamente con los otros jugadores que estén conectados a la red.

Donde lee un archivo por consola y cada simulación está separada por un renglón en el archivo, donde cada renglón contiene los números a multiplicar separados por espacio. Cada renglón es una simulación diferente con un  $t$  (grado del polinomio) diferente.

## Instrucciones para correr la red

Antes de ejecutar el programa, asegúrate de contar con los siguientes elementos:

### 1 Requisitos Previos

Antes de ejecutar el programa, asegúrate de contar con los siguientes elementos:

- Python 3.9 o superior instalado en tu sistema.
- **Módulos de Python:** Asegúrate de tener instalados los siguientes módulos de Python:
  - **ssl:** Para manejar conexiones seguras.
  - **socket:** Para manejar conexiones de red.
  - **threading:** Para manejar múltiples conexiones simultáneamente.
  - **uuid:** Para generar identificadores únicos.
  - **json:** Para manejar archivos de configuración en formato JSON.

Puedes instalar los módulos necesarios utilizando pip:

```
pip install ssl socket threading uuid json
```

- Un archivo JSON con la configuración de conexiones.
- Un certificado SSL (`cert.pem`) y una clave privada (`key.pem`) para la comunicación segura.

### 2 Configuración del Archivo de Conexiones

El programa utiliza un archivo JSON para configurar los usuarios y su conexión.

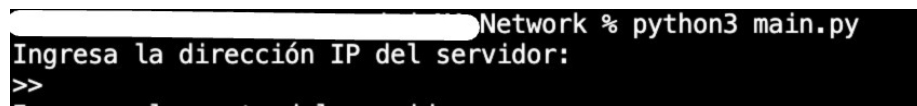
Guarda este archivo JSON con un nombre como **connections.json**.

- **host:** Define la configuración del host principal, incluyendo la dirección IP, el puerto y un identificador único (UUID).

- **users:** Define una lista de usuarios con los que el host se conectará. Cada usuario tiene una dirección IP, un puerto y una lista de números que se utilizarán en el programa.

### 3 Ejecución del Programa individualmente

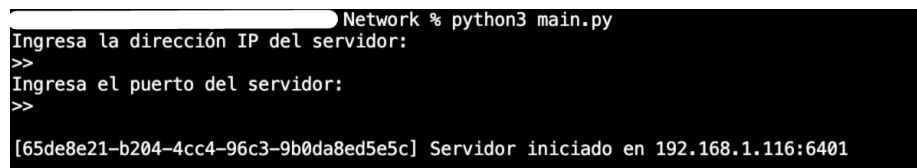
Tendremos que ir a la carpeta **Network** y correr el archivo **main.py** para correr el protocolo en la terminal. Cuando digitemos el comando **python3 main.py** en la terminal, se nos pedira que ingresemos el la dirección ip de nuestro dispositivo, si no se tiene solo damos Enter.



```
Network % python3 main.py
Ingresa la dirección IP del servidor:
>>
```

Figure 3: Solo enter y nada mas.

Despues nos preguntara nuestro puerto, si no se tiene solo damos Enter. al igual que con la direccion ip. Asi el programa se ejecutara y nos dira la ip y el puerto que se esta utilizando.



```
Network % python3 main.py
Ingresa la dirección IP del servidor:
>>
Ingresa el puerto del servidor:
>>
[65de8e21-b204-4cc4-96c3-9b0da8ed5e5c] Servidor iniciado en 192.168.1.116:6401
```

Figure 4: Solo enter y nada mas.

damos help para conocer los comandos que podemos utilizar en la terminal. en estos apareceran los comandos:

- **connect:** Conectarnos con otros usuarios.
- **message:** Para enviar un mensaje a otro usuario.
- **number:** Para enviar el numero que multiplicaremos.
- **multiply:** Para iniciar la multiplicación.
- **reconstruct:** Para reconstruir el resultado.
- **status:** Para saber con cuantos estamos conectados y los fragmentos que tenemos.
- **exit:** Para salir del programa.

```
>> help
Comandos disponibles:
- connect
- message
- number
- multiply
- reconstruct
- status
- exit
>> connect 192.1[REDACTED]
```

Figure 5: Despues de cada comando va un espacio

Luego daremos connect, espacio y escribiremo la ip del usuario al que nos queremos conectar. si el usuario esta conectado a alguien mas, nos deberiamos conectar con el tambien. Si esto no pasa y el algoritmo de conexion falla. Debemos conectarnos a los otros usuarios manualmente uno por uno.

Podemos colocar status para ver con cuantos usuarios estamos conectados y los fragmentos que tenemos, y varias cosas mas.

```
>> connect 192.1[REDACTED] 6 5835
[65de8e21-b204-4cc4-96c3-9b0da8ed5e5c] Enviando solicitud de conexión a 192.168.1.116:5835
>>
[65de8e21-b204-4cc4-96c3-9b0da8ed5e5c] Conexión establecida con 97a409a0-9c80-4785-8f7e-9b0c65e855f7 | 192.168.1.116:5835
connect 192.168.1.116 5973
[65de8e21-b204-4cc4-96c3-9b0da8ed5e5c] Enviando solicitud de conexión a 192.168.1.116:5973
>>
[65de8e21-b204-4cc4-96c3-9b0da8ed5e5c] Conexión establecida con 2569c26b-3df7-4ceb-8d4b-c9bb90fef92d | 192.168.1.116:5973
status
Usuarios conectados:
- 2569c26b-3df7-4ceb-8d4b-c9bb90fef92d (192.168.1.116:5973)
- 65de8e21-b204-4cc4-96c3-9b0da8ed5e5c (192.168.1.116:6401)
- 97a409a0-9c80-4785-8f7e-9b0c65e855f7 (192.168.1.116:5835)
Partes:
Operaciones:
Resultados:
Final Shares:
```

Figure 6: El numero al que apunta la linea roja es el puerto al que nos conectaremos.

Luego de conectarnos con todos los usuarios, escribiremos el comando number, espacio y el numero que queremos enviar a los otros usuarios. Todos los usuarios lo deben hacer simultáneamente y Debemos esperar a que todos los usuarios envíen el número antes de continuar.

luego de esto, escribiremos el comando multiply para iniciar la multipli-

cación. Todos los usuarios deben hacerlo simultáneamente. y despues nos dira que recibimos que hemos recibido los fragmentos y podremos hacer la Reconstrucción.

```
>> number 9
Número enviado.
>> multiply
Operación enviada.
>>
[65de8e21-b204-4cc4-96c3-9b0da8ed5e5c] Recibida parte final de 97a409a0-9c80-4785-8f7e-9b0c65e855f7
[65de8e21-b204-4cc4-96c3-9b0da8ed5e5c] Recibida parte final de 2569c26b-3df7-4ceb-8d4b-c9bb90fef92d
[65de8e21-b204-4cc4-96c3-9b0da8ed5e5c] Recibida parte final de 65de8e21-b204-4cc4-96c3-9b0da8ed5e5c
```

Figure 7: Podemos observar la encriptación de los mensajes por la libreria ssl.

luego procedemos a escribir el comando reconstruct para reconstruir el resultado. y el resultado se mostrara en la terminal.

```
reconstruct
Secreto reconstruido: 81
```

Figure 8: El resultado de la multiplicación es 81.

## 4 Ejecución del Programa en Red Local dentro del mismo dispositivo

### Paso 1: Inicializar el Host Principal

Para iniciar el host principal, crea un objeto `ConnectionsFile` y un `MainUser`:

```
from ConnectionsFile import ConnectionsFile

import NetworkUser

config = ConnectionsFile("connections.json") # Carga el archivo JSON

host = config.create_host() # Crea el usuario principal
```

### Paso 2: Conectar con los Usuarios

Luego, conecta el host con los usuarios definidos en el archivo JSON:

```
config.connect_with_users(host)
```

El programa intentará conectar con cada usuario definido en la lista `users`. Si hay algún problema, mostrará mensajes de error.

## 5 Funcionalidad del Programa

### Protocolos de Comunicación

El programa maneja varios protocolos para la transmisión de datos entre los usuarios de la red:

1. `REQUEST_CONNECTION`: Solicita una conexión a otro usuario.
2. `ACCEPT_CONNECTION`: Acepta una conexión entrante.
3. `MESSAGE`: Envía un mensaje de texto entre usuarios.
4. `INPUT_SHARE`: Comparte un valor secreto utilizando el protocolo de Shamir.
5. `PRODUCT_SHARE`: Comparte el resultado de una operación de multiplicación entre valores secretos.
6. `FINAL_SHARE`: Envía la parte final de un cálculo entre los usuarios conectados.

Cada protocolo tiene funciones `send_message()` y `receive_message()` para manejar el envío y recepción de datos.

### Envío de Datos Secretos

Para enviar un número secreto a los usuarios conectados, se usa la función:

```
host.send_number(42) # Envía el número 42 a los usuarios usando el protocolo de Shamir
```

También se puede especificar un protocolo específico:

```
from NetworkProtocol import FinalShareProtocol
host.send_number(99, FinalShareProtocol)
```

### Reconstrucción del Secreto

Una vez recibidas todas las partes necesarias, un usuario puede reconstruir el secreto:

```
secreto_reconstruido = host.reconstruct_secret()
print(f'Secreto final: {secreto_reconstruido}')
```



## 6 Posibles Errores y Soluciones

Si encuentras algún problema al ejecutar el programa, revisa la siguiente tabla con los errores más comunes y sus posibles soluciones:

Error	Posible Causa	Solución
Error al leer el archivo JSON	Archivo <code>connections.json</code> mal estructurado o inexistente.	Verifica la sintaxis JSON y que el archivo exista en el directorio correcto.
No se ha definido el <code>host</code> .	El archivo JSON no tiene una sección <code>host</code> válida.	Revisa que la sección <code>host</code> en el JSON contenga <code>ip</code> , <code>port</code> y <code>uuid</code> .
No se pudo conectar con <code>&lt;IP&gt;:&lt;Puerto&gt;</code>	El usuario no está en línea o el puerto está cerrado.	Asegúrate de que el usuario remoto esté ejecutando el programa y que el puerto esté disponible.
Se agotaron los intentos de reconexión.	Problema de conexión persistente.	Verifica la configuración de red, el firewall y que el host remoto esté accesible.