Proyecto de redes de computadoras

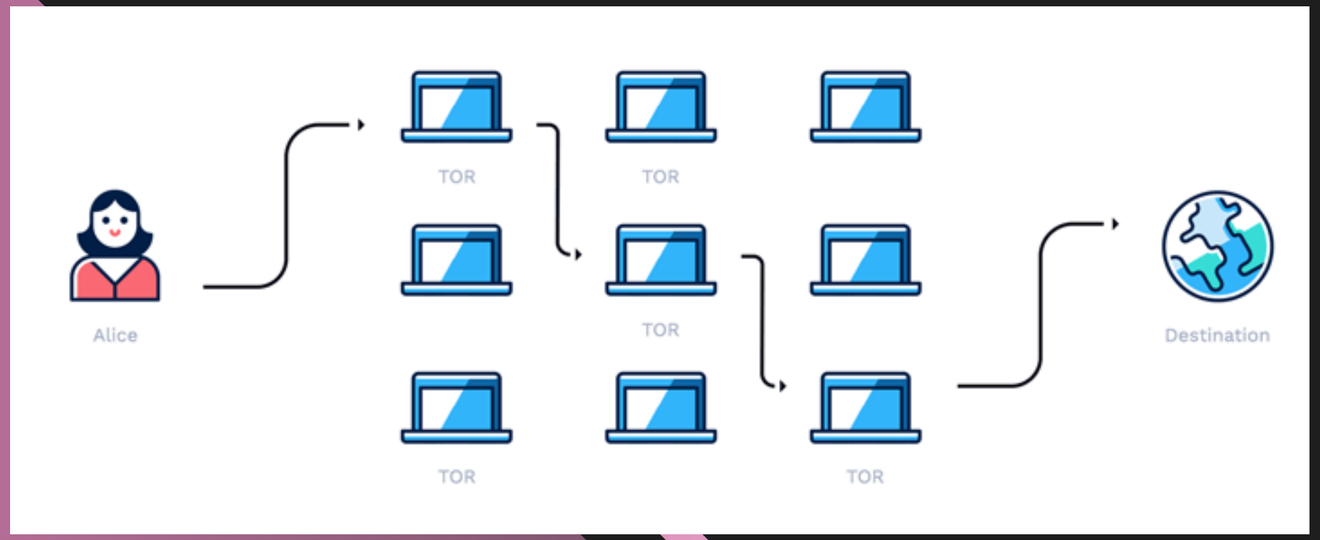
RED TOR

Simulación

Presentado por:

Alex Perez, Nicolas Moina

¿Qué es TOR?



Tor, que significa "The Onion Router", es una red que permite a los usuarios navegar y comunicarse de forma anónima. Esta usa una técnica de enrutamiento por capas para cifrar y redirigir el tráfico de internet mediante distintos servidores voluntarios. De esta forma se dificulta el rastreo del origen del tráfico o la identificación de los usuarios. Esta red es usada para proteger la privacidad personal y evadir la censura.

Objetivos

01

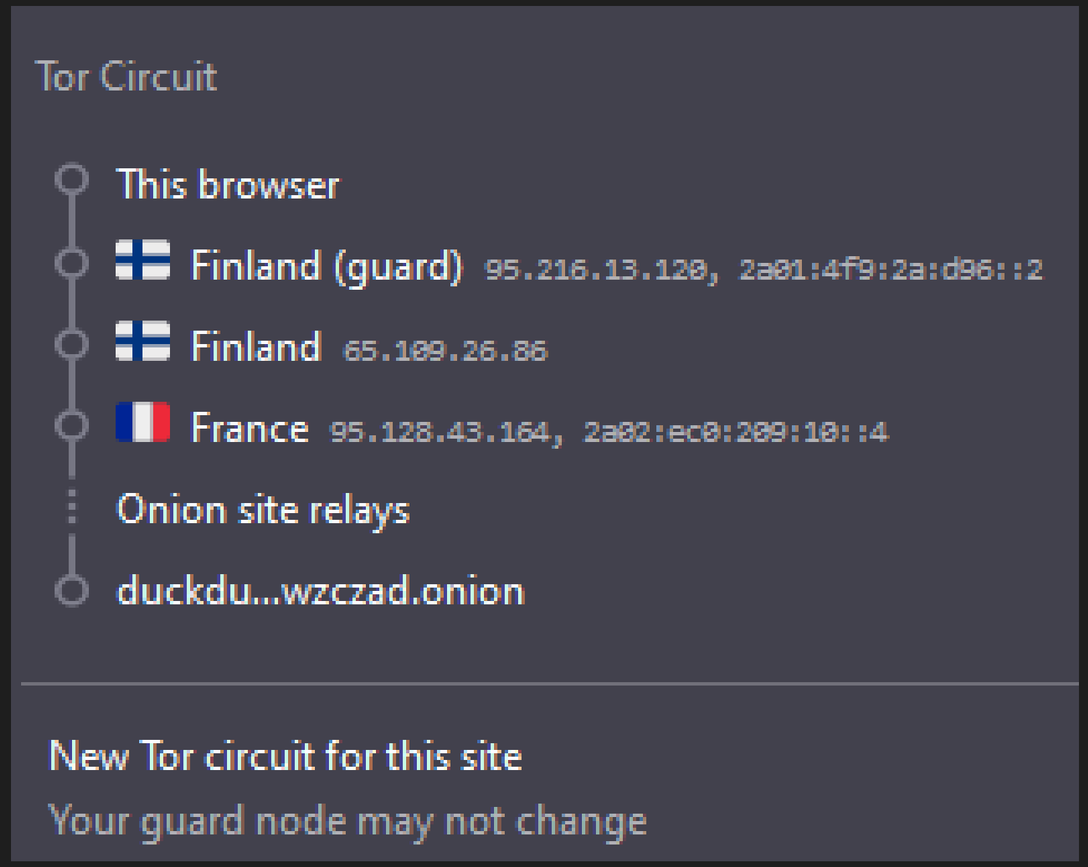
Investigar y entender el funcionamiento del protocolo Tor, incluyendo su estructura, encriptación de capas, y selección de rutas.

02

Desarrollar una versión simplificada del protocolo que emule el proceso de encriptación en capas y transmisión de datos a través de relays seleccionados.

Arquitectura TOR

- Cliente Tor: El software que el usuario instala en su dispositivo. Configura la conexión inicial y maneja la encriptación y desencriptación de datos.
- Nodos de Tor: Servidores voluntarios que forman la red Tor.
  - Guard Nodes (Nodos de Guardia): El primer nodo con el que un cliente se conecta. Estos nodos son los únicos que conocen la dirección IP real del usuario.
  - Nodos Intermedios: Encaminan el tráfico de Tor sin conocer ni el origen ni el destino final del mismo. Proporcionan un nivel adicional de separación entre el usuario y el destino.
  - Nodos de Salida: El último nodo en la cadena de Tor antes de que el tráfico alcance su destino final en Internet abierta. Este nodo desencripta la última capa de encriptación y envía el tráfico a su destino final.
- Servidores de Directorio: Mantienen una lista de todos los nodos activos y su estado. Ayudan a los clientes a conocer qué nodos están disponibles para crear un circuito.



Nuestra Simulación

FLASK

Herramienta de Python que ayuda a crear aplicaciones web. Se utiliza para manejar las comunicaciones HTTP, lo que facilita el intercambio de datos entre los diferentes nodos de la red.

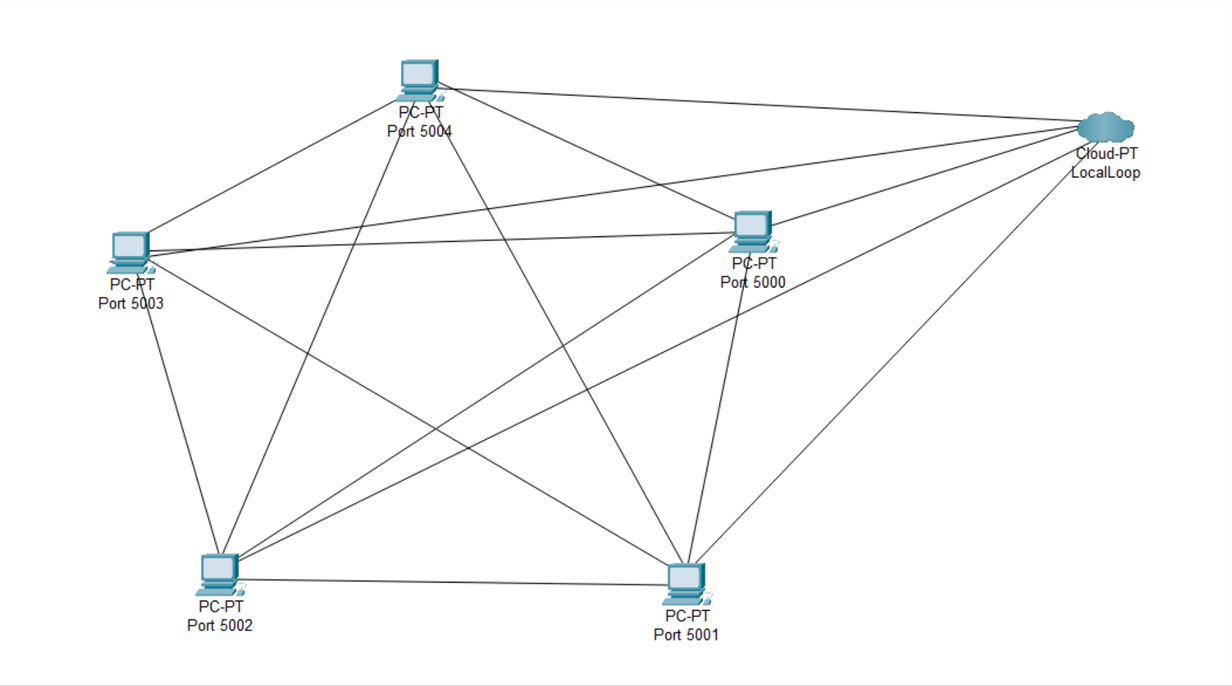
CRYPTOGRAPHY

Librería que proporciona herramientas para la encriptación y desencriptación de mensajes, simulando la seguridad de comunicación entre nodos de la red TOR.

POSTMAN

Aplicación utilizada para enviar solicitudes HTTP POST a Flask. Estas solicitudes emulan el comportamiento de un nodo en la red que intenta enviar un mensaje cifrado a otro nodo. Se configura para enviar un cuerpo de mensaje (payload), que en el entorno real sería un mensaje cifrado, al nodo destino

Nuestra Simulación



Los mensajes pasan a través de varios puntos o nodos, pero cada nodo sólo conoce el punto inmediatamente anterior y el siguiente en la ruta, sin saber el origen inicial o el destino final del mensaje. Esto ayuda a ocultar la identidad tanto del remitente como del destinatario.

Funcionamiento

GENERACIÓN Y ENCRIPTACIÓN

Cuando se crea un mensaje, este se cifra usando la técnica RSA, junto con un método adicional llamado OAEP que añade una capa extra de seguridad. Esto se hace utilizando la "clave pública" del destinatario, similar a una cerradura que solo el nodo correcto puede abrir.

TRANSMISIÓN Y DESENCRIPTACIÓN

Una vez cifrado, el mensaje viaja por diferentes nodos. Ningún nodo de tránsito puede leer el mensaje porque está cifrado. Cuando el mensaje llega a su nodo destino, se descifra usando una "clave privada" que solo tiene dicho nodo, asegurando que solo él pueda leerlo.

FLAKS

Usado para definir rutas específicas que funcionan como endpoints para manejar el envío y recepción de mensajes cifrados. El endpoint /send, procesa solicitudes HTTP POST para recibir un mensaje cifrado de un nodo, procesarlo, y luego redirigirlo al siguiente nodo designado mediante otra solicitud POST. Flask también maneja internamente las rutas que deben seguir los mensajes, utilizando variables de entorno y configuraciones para determinar dinámicamente a qué nodo deben dirigirse los mensajes según la información incluida en cada solicitud. Se configuró para simular un local loop, lo que permite que los mensajes se envíen y reciban dentro del mismo sistema sin necesidad de una red externa

- Cada nodo tiene su propio conjunto de claves almacenadas de forma segura en archivos. Esto garantiza que los mensajes sólo puedan ser leídos por los nodos a quienes están destinados.
- Los mensajes se envían a través de varios nodos y cada uno solo sabe de dónde vino el mensaje y hacia dónde va después.
- Se usa un sistema de cifrado llamado RSA y se ecogen los caminos que los mensajes toman de manera aleatoria.

Capturas

Nodo 0: Puerto 5000 (Guard)

```
Nodo 0 enviando mensaje a nodo 1
Mensaje encriptado -----> Bgcs6rAUSl868ivE6YYCeo03inMrI7Ux7do2dU5JLajcQU8uLEJe5wYlhxS3MXaWnZ1NFiGnXuCSkntu5RwTcBY
wfrSVFZbh1noB3zd+djWG8TTocJqNjYBOuHz+hclGIl0rulg6qE56pp5vO06JtxpBAbY37SzYJqqKkYzRKcW2fIjbxxziE7Loo3Yt7wF9rZtTe9OE
tktfhsyDYqtQwc4qutcLUYbf17BqbWYj5IO1enflDtlCrHwvkFBpMsbYR1d6pcRVibbfa/XE0YYFdh5XUBb5EK6TYw/A6zkc1KTGCwsPgy2b2P3+0
tUjldae0kb0hsMLlkyKLs0UjsoG0w== type <class 'str'>
127.0.0.1 - - [05/May/2024 14:07:21] "POST /send HTTP/1.1" 200 -
```

Nodo 1: Puerto 5001

```
Mensaje encriptado -----> Bgcs6rAUSl868ivE6YYCeo03inMrI7Ux7do2dU5JLajcQU8uLEJe5wYlhxS3MXaWnZ1NFiGnXuCSkntu5RwTcBY
wfrSVFZbh1noB3zd+djWG8TTocJqNjYBOuHz+hclGIl0rulg6qE56pp5vO06JtxpBAbY37SzYJqqKkYzRKcW2fIjbxxziE7Loo3Yt7wF9rZtTe9OE
tktfhsyDYqtQwc4qutcLUYbf17BqbWYj5IO1enflDtlCrHwvkFBpMsbYR1d6pcRVibbfa/XE0YYFdh5XUBb5EK6TYw/A6zkc1KTGCwsPgy2b2P3+0
tUjldae0kb0hsMLlkyKLs0UjsoG0w==
Nodo 1 enviando mensaje a nodo 2
127.0.0.1 - - [05/May/2024 14:07:21] "POST /send HTTP/1.1" 200 -
```

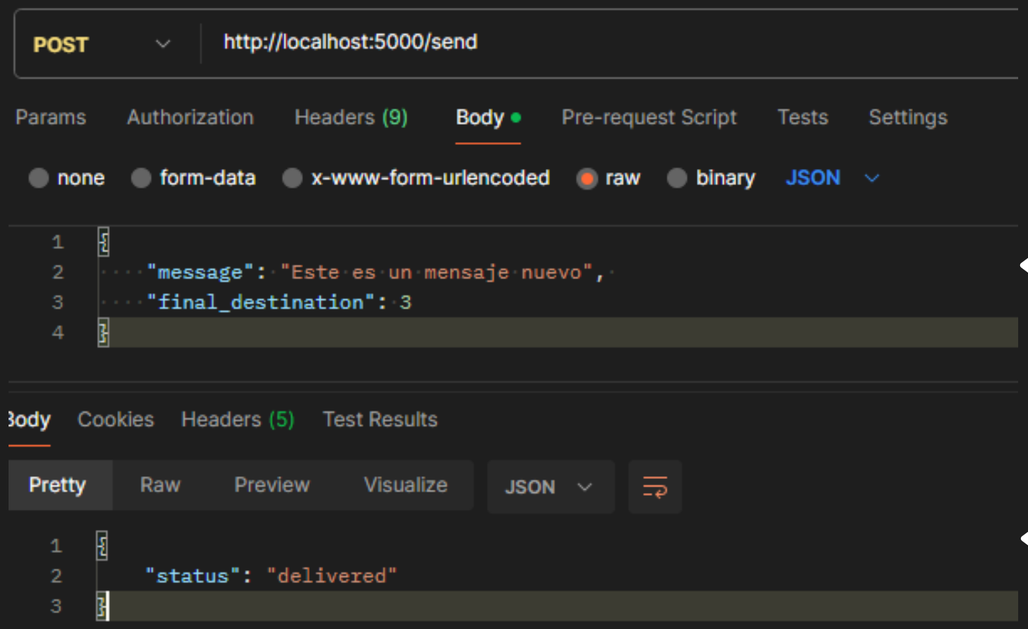
Nodo 2: Puerto 5002

```
Mensaje encriptado -----> Bgcs6rAUSl868ivE6YYCeo03inMrI7Ux7do2dU5JLajcQU8uLEJe5wYlhxS3MXaWnZ1NFiGnXuCSkntu5RwTcBY
wfrSVFZbh1noB3zd+djWG8TTocJqNjYBOuHz+hclGIl0rulg6qE56pp5vO06JtxpBAbY37SzYJqqKkYzRKcW2fIjbxxziE7Loo3Yt7wF9rZtTe9OE
tktfhsyDYqtQwc4qutcLUYbf17BqbWYj5IO1enflDtlCrHwvkFBpMsbYR1d6pcRVibbfa/XE0YYFdh5XUBb5EK6TYw/A6zkc1KTGCwsPgy2b2P3+0
tUjldae0kb0hsMLlkyKLs0UjsoG0w==
Nodo 2 enviando mensaje a nodo 3
127.0.0.1 - - [05/May/2024 14:07:21] "POST /send HTTP/1.1" 200 -
```

Nodo 3: Puerto 5003(Salida)

```
Mensaje encriptado -----> Bgcs6rAUSl868ivE6YYCeo03inMrI7Ux7do2dU5JLajcQU8uLEJe5wYlhxS3MXaWnZ1NFiGnXuCSkntu5RwTcBY
wfrSVFZbh1noB3zd+djWG8TTocJqNjYBOuHz+hclGIl0rulg6qE56pp5vO06JtxpBAbY37SzYJqqKkYzRKcW2fIjbxxziE7Loo3Yt7wF9rZtTe9OE
tktfhsyDYqtQwc4qutcLUYbf17BqbWYj5IO1enflDtlCrHwvkFBpMsbYR1d6pcRVibbfa/XE0YYFdh5XUBb5EK6TYw/A6zkc1KTGCwsPgy2b2P3+0
tUjldae0kb0hsMLlkyKLs0UjsoG0w==
Mensaje final en destino -----> Bgcs6rAUSl868ivE6YYCeo03inMrI7Ux7do2dU5JLajcQU8uLEJe5wYlhxS3MXaWnZ1NFiGnXuCSkntu5
RwTcBYwfrSVFZbh1noB3zd+djWG8TTocJqNjYBOuHz+hclGIl0rulg6qE56pp5vO06JtxpBAbY37SzYJqqKkYzRKcW2fIjbxxziE7Loo3Yt7wF9rZ
tTe9OEtktfhsyDYqtQwc4qutcLUYbf17BqbWYj5IO1enflDtlCrHwvkFBpMsbYR1d6pcRVibbfa/XE0YYFdh5XUBb5EK6TYw/A6zkc1KTGCwsPgy2
b2P3+0tUjldae0kb0hsMLlkyKLs0UjsoG0w==
Mensaje recibido en destino final 3: Este es un mensaje nuevo
127.0.0.1 - - [05/May/2024 14:07:21] "POST /send HTTP/1.1" 200 -
```

Postman Captura



Payload

Response

Mensaje no encriptado

```
Frame 926: 379 bytes on wire (3032 bits), 379 bytes captured (3032 bits) on interface \Device\NPF_Loopba
Null/Loopback
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 59931, Dst Port: 5000, Seq: 1, Ack: 1, Len: 335
Hypertext Transfer Protocol
JavaScript Object Notation: application/json
  Object
    Member: message
        [Path with value: /message:Este es un mensaje nuevo]
        [Member with value: message:Este es un mensaje nuevo]
        String value: Este es un mensaje nuevo
        Key: message
        [Path: /message]
    Member: final_destination
        [Path with value: /final_destination:3]
        [Member with value: final_destination:3]
        Number value: 3
        Key: final_destination
        [Path: /final_destination]
```

WireShark (LocalLoop)

Mensaje encriptado punto a punto

```
Transmission Control Protocol, Src Port: 59933, Dst Port: 5002, Seq: 204, Ack: 1, Len: 396
[2 Reassembled TCP Segments (599 bytes): #1125(203), #1127(396)]
Hypertext Transfer Protocol
JavaScript Object Notation: application/json
  Object
    Member: message
        [Path with value [truncated]: /message:WejKXIthnoneEg2iKuat4u7EcBHRQp78RrfZ9/rTnaLz89h
        [Member with value [truncated]: message:WejKXIthnoneEg2iKuat4u7EcBHRQp78RrfZ9/rTnaLz89
        String value [truncated]: WejKXIthnoneEg2iKuat4u7EcBHRQp78RrfZ9/rTnaLz89hRk14GDFdva5tg
        Key: message
        [Path: /message]
    Member: final_destination
        [Path with value: /final_destination:3]
        [Member with value: final_destination:3]
        Number value: 3
        Key: final_destination
        [Path: /final_destination]
    Member: path
      Array
          [Path with value: /path/[]:0]
          [Member with value: []:0]
          Number value: 0
```

Desafíos

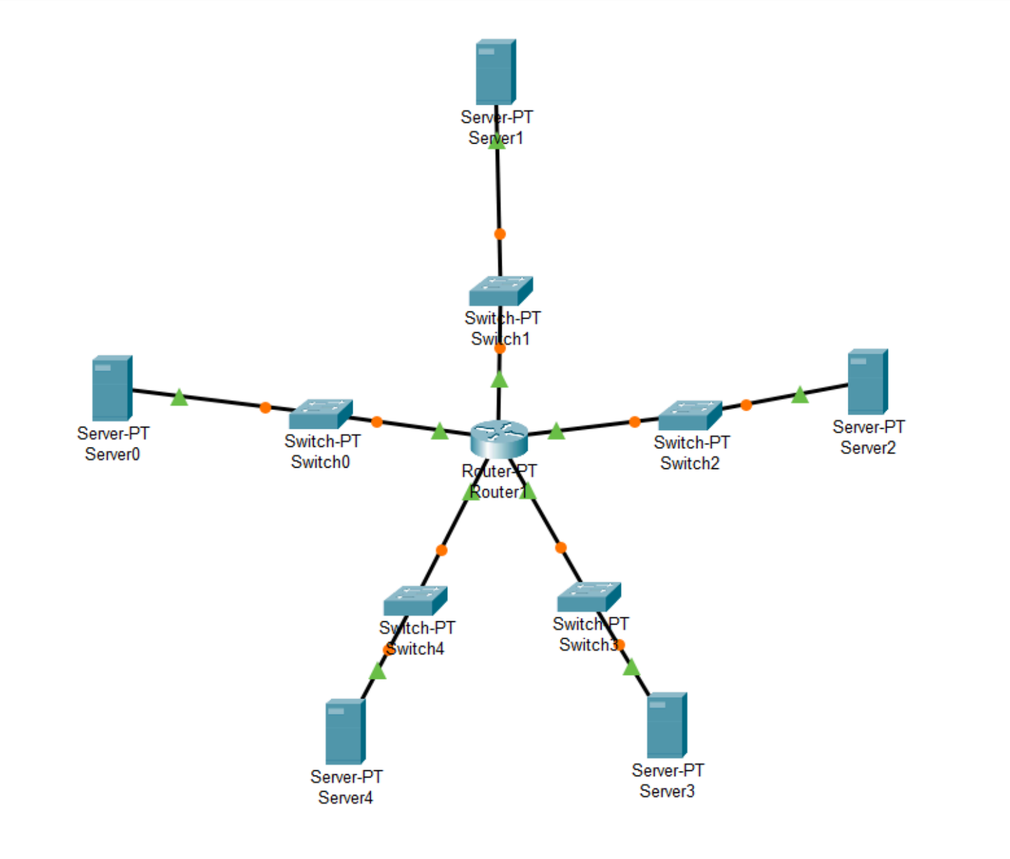
01 Implementación de claves públicas / privadas

02 Creación de servidores

03 Creación de rutas aleatorias

Implementaciones futuras

- Mejora de la Escalabilidad: Para hacer nuestro prototipo más realista y robusto, planeamos incrementar el número de nodos en la red y mejorar la eficiencia del enrutamiento.
- Implementación de Características Adicionales de Tor: Integrar funcionalidades que no se incluyeron en la versión inicial, como los puentes Tor y los nodos de salida más seguros, para simular de manera más precisa la red Tor.
- Implementacion de una interfaz de usuario para que sea mucho mas amigable



¿Preguntas?

MUCHAS GRACIAS