Trabajo final

Este trabajo representa una situación dónde un cliente necesita recibir un mensaje desde un servidor pero de forma segura. Para ello, el servidor cifrará el mensaje con una clave compartida entre el cliente y el servidor. Esta clave compartida será el resultado de la aplicación del algoritmo Diffie-Hellman entre el cliente y el servidor.

El objetivo es desarrollar un cliente en Python 3 que inicialmente implemente el algoritmo de Diffie-Hellman intercambiando mensajes por UDP con el servidor. Una vez finalizado el intercambio, ambas entidades cliente y servidor deben haber calculado una clave compartida común. A continuación el servidor cifrará el mensaje usando esta clave compartida y lo enviará cifrado al cliente mediante TCP. El cliente descifrará este mensaje usando la misma clave compartida que debe haber calculado e imprimirá por pantalla el mensaje descifrado.

El mecanismo básico puede ser descrito como:

FUNCIONAMIENTO DEL SERVIDOR

I. La ejecución del servidor no llevará ningún parámetro adicional.
II. Pondrá a la escucha el puerto UDP 45.000
III. A continuación se implementa el algoritmo Diffie-Hellman.
    - Por este puerto recibe un mensaje conteniendo 3 valores enteros expresados como dígitos separados por comas sin espacios. Estos valores son A (número calculado por el cliente), p (número primo: entre 23 y 50) y g (número generador: aleatorio entre 1 y 7)
    - Calcula un número aleatorio b (entre 1 y 20)
    - Calcula el número *B=g^b mod p*
    - Envía al cliente por UDP, al mismo puerto origen del mensaje anterior, el valor calculado de B como un entero expresado en dígitos como cadena de caracteres.
    - Calcula la clave compartida como *K=A^b mod p*
IV. Una vez obtenida la clave compartida el servidor espera 5 segundos antes de proceder
V. Inicia una conexión TCP al cliente a su puerto 50.000 que estará a la escucha.
VI. Una vez establecida la conexión, el servidor envía un mensaje cifrado usando como clave el valor de K calculado anteriormente.
VII. Una vez enviado el mensaje espera durante 5 segundos la recepción de un mensaje por parte del cliente con el contenido "*OK*".
VIII. Tanto si recibe el mensaje "OK" como si no, a los 5 segundos corta la conexión TCP y vuelve a empezar la escucha inicial por UDP

FUNCIONAMIENTO DEL CLIENTE

IX. La ejecución del cliente lleva como único parámetro la IP del servidor
X. Se inicia el algoritmo Diffie-Hellman
    - Calcula los valores: a (número aleatorio entre 1 y 20), p (número primo entre 23 y 50 -seleccionarlo manualmente-) y g (número generador: aleatorio entre 1 y 7)

- Calcula el valor de A: *A=g^a mod p*
- Envía por UDP al puerto 45.000 del servidor una cadena de caracteres formada por 3 valores enteros expresados en dígitos separados por comas sin espacios. Este mensaje está compuesto por los valores *A,p y g*
- Recibe por UDP desde el servidor un mensaje textual con un valor entero expresado en dígitos conteniendo el valor B calculado por el servidor
- Calcula la clave compartida *K=B^a mod p*

XI. Una vez calculada la clave compartida para descifrar el mensaje, el cliente pone a la escucha el puerto TCP número 50.000. Por este puerto recibirá la petición de conexión por parte del servidor y la aceptará

XII. A través de la conexión TCP establecida, recibe un mensaje cifrado por parte del servidor de tamaño contante 80 bytes

XIII. Contestará enviando un mensaje TCP al servidor con la cadena "*OK*"

XIV. Descifrará el mensaje recibido usando la clave K e imprimirá por pantalla el mensaje descifrado

XV. El cliente cierra todos los sockets y termina su ejecución

La implementación

En un nivel más detallado, el cliente que implementes debe realizar los siguientes pasos (en este orden):

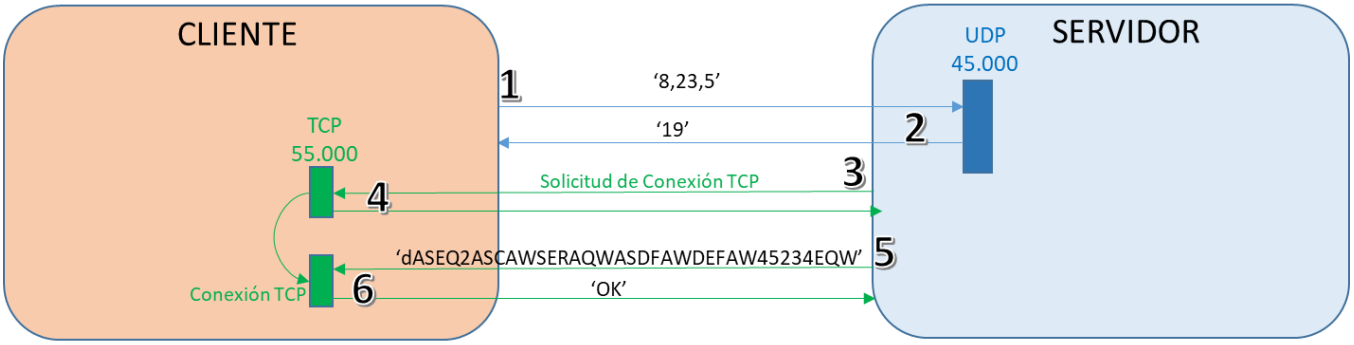
(a) Si el usuario escribe más de dos argumentos en línea de comandos (i.e. *sys.argv[0]*, *sys.argv[1]*) se muestra un error indicando "*Error. Uso: cliente IP*" y se sale del programa.
(b) Realiza los cálculos iniciales del algoritmo Diffie-Hellman
(c) Imprime por pantalla los valores a,p,g y A. Por ejemplo:
a=6, p=23, g=5, A=8
(d) Envía por UDP el mensaje al servidor a su puerto 45.000. Por ejemplo: "*8,23,5*"
(e) Recibe por UDP el mensaje del servidor desde el puerto 45.000. Por ejemplo: "*19*"
(f) Calcula la clave compartida K y la imprime por pantalla. Por ejemplo:
K=2
(g) Pone a la escucha el puerto TCP número 50.000
(h) Acepta la petición de conexión del servidor
(i) Recibe por la conexión TCP establecida un mensaje cifrado
(j) Envía por la conexión TCP el mensaje "OK"
(k) Descifra el mensaje usando la clave de descifrado que ha obtenido al usar la K calculada como parámetro (Vea las notas al final)
(l) Imprime por pantalla el mensaje descifrado
(m) Cierra todos los sockets y finaliza el programa

Un ejemplo de ejecución del cliente y del servidor sería:

ejecución del cliente solo registro	ejecución del servidor

%>python3 cliente.py 127.0.0.1	%>python3 server.pyc

A título de ejemplo, en la gráfica adjunta se muestra la secuencia de diálogo UDP y TCP entre el cliente y el servidor:



Tratamiento de Errores

Errores que pueden ocurrir: (los que no se listen no se tienen por qué considerar y no se probarán)

- número de argumentos en línea de comandos erróneo.
- errores de sockets (p.ej. conexión no realizada, etc..)

Notas:

- Las fórmulas a usar en Python son muy sencillas. Por ejemplo *A=g^a mod p* Sería:

```
A = (g ** a) % p
```

- Para descifrar el mensaje utilice el algoritmo TRIPLEDES que es el que utilizará el servidor, para ello instale mediante pip el paquete `pyDes`. Una vez instalado puede usar sus métodos tras importarlos

```
from pyDes import *

texto_descifrado = triple_des(obtiene_clave_cifrado(K)).decrypt('Mensaje
cifrado',padmode=2)
```

- Cálculo de la clave de descifrado.
  La clave de cifrado/descifrado debe ser una cadena de 16 bytes de longitud obtenida a partir del valor K resultado del cálculo de la clave compartida. Puede usar esta función para obtenerla usando como parámetro la K calculada, importando previamente el paquete `hashlib`

```
def obtiene_clave_cifrado(K):
    m = hashlib.md5(str(K).encode('utf-8'))
    return m.hexdigest()[:16]
```

Soporte:

Para más información acerca del algoritmo de Diffie-Hellman puede consultar el enlace:
https://es.wikipedia.org/wiki/Diffie-Hellman

Se ha habilitado un foro en la web de enseñanza virtual para plantear y resolver dudas respecto a este trabajo de curso