Workspace for '20221-transportes-caso3'

Page 1 (row 1, column 1)

Server

que guarda un nombre de cliente, un identificador de paquete y el estado de cada paquete recogido

→ la famosa tabla →

PKT-ID	USR	STATUS

responde las consultas de los clientes.

Como funcionan las consultas →

Para consultar el estado de un paquete, un cliente envía al servidor su nombre y el identificador de un paquete, el servidor recibe el identificador, consulta la información guardada y responde con el estado correspondiente

↓ si no existe usuario ID

el servidor retorna el estado "DESCONOCIDO"

- lo unico que responde el servidor es el estado del paquete

↳ Esto sera el mensaje que tiene que encriptar tambien

El servidor debe medir el tiempo que toma cifrar el reto con su llave privada, y el tiempo que toma cifrar ese mismo reto con la llave simétrica compartida.

Estados de un paquete →

(PKT_EN_OFICINA, PKT_RECOGIDO, PKT_EN_CLASIFICACION, PKT_DESPACHADO, PKT_EN_ENTREGA, PKT_ENTREGADO, PKT_DESCONOCIDO)

Condiciones generales

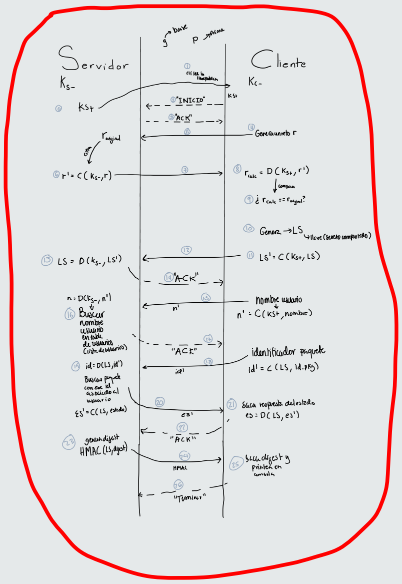
Las comunicaciones entre cliente-servidor deben usar sockets y deben estar cifradas.

(java.security y javax.crypto).

→ toca investigar

Generaremos por adelantado las llaves pública y privada del servidor.

Sistema Completo



Cliente

Para consultar el estado de un paquete, un cliente envía al servidor su nombre y el identificador de un paquete, el servidor recibe el identificador, consulta la información guardada y responde con el estado correspondiente.

↓

El cliente hace requests con PKG.ID y USR.ID

↳ Estas tendran que estar encriptadas

Es un programa que envía una consulta al servidor, espera la respuesta y al recibirla la valida.

¿Que hace con las respuestas? →

Si la respuesta pasa el chequeo entonces despliega la repuesta en pantalla, si no pasa el chequeo entonces despliega el mensaje "Error en la consulta".

↳ El cliente tambien tiene que estar listo para recibir informacion de volta entonces siempre tiene que estar oyendo.

El cliente lee la llave pública del servidor de un archivo. Es decir, suponemos que el cliente previamente obtuvo la llave pública del servidor.

↓

El cliente ya tiene guardada la Ks+ por default.

↓

Maybe la lee de un archivo o maybe ya es como una constante.

Q1:

Para simplificar el problema, tendremos un servidor con una tabla predefinida con nombre de usuario, identificador de paquete y estado de 32 paquetes

- OK esto como seria. son 32 entradas en total o es como numero variable de usuarios (upto 32 o más?) y 32 paquetes fijos?

- Seria como una tabla asi?

identificadorPaquete	usuario	estado
pk1	usr 1	PKT_EN_OFICINA
pk2	usr 2	PKT_RECOGIDO
pk3	usr 5	~~~
⋮	⋮	~~~
pk 32	usr 3	~~~

Usuarios: usr 1, usr 2, ⋮, usrn

id Paquetes: pck 1, ⋮, pck 32 → Solo 32 paquetes

estados: xxxx

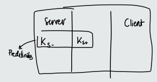
Q2:

- En la tabla de paquetes son los id de paquetes unicos?
- Puede un usuario tener multiples paquetes?

Q3:

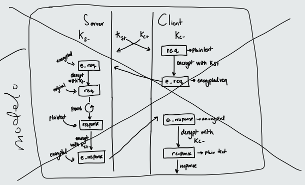
Generaremos por adelantado las llaves pública y privada del servidor.

- Asumo que esto es un esquema de public key (asymetric)
- Es algo como:



- En este caso toca que cada cliente cree su clave publica y privada (Kc+, Kc-) o no?

- La vaina seria como:



Leyendo el protocolo el main com es simetrico

- Como voy a generar las llaves en general?
- Como voy a compartir las llaves publicas? → toca hacer como otro protocolo previo? se guarda en alguna tabla de uso comun?

Q4:

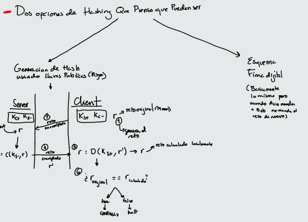
El cliente envía al servidor un reto (un número aleatorio de 24 dígitos).

El servidor responde con el reto cifrado con su llave privada.

Al recibir la respuesta, el cliente valida que el reto cifrado tenga el valor esperado; si la validación pasa entonces el cliente continúa con el protocolo; si la validación no pasa entonces el cliente termina la comunicación

- Eso de reto a que concepto de clase se acerca?
- ¿Es esto basicamente lo de hashing para asegurar integridad?

- Dos opciones de Hashing Que Pienso que Pueden ser


Generacion de Hash usando llaves Publicas (Asym)
Server
Kc+ Ks-
plaintext
r
③ r' = C(ks-, r)
② reto no encriptado
④ reto encriptado r'
Client
Ks+ Kc-
r → reto original (24 num)
① genera el reto
⑤ r = D(Ks+, r') → r → reto calculado localmente
⑥ ¿ r original == r calculado?
true → continuar
false → halt
Esquema Firma digital
(Basicamente lo mismo pero cuando Alice manda a Bob no manda el reto de nuevo)

Q5:

El servidor envía un código de resumen o digest.

- En que consiste el digest?

El cliente debe validar la integridad de la información recibida

- Valida la info localmente o como hace con su vida?

Q6:

Corra su programa en diferentes escenarios y mida el tiempo que el servidor requiere para cifrar el reto con cifrado simétrico y con cifrado asimétrico.

- Tacho. Tal como lo explican con la verificacion de la cifra el modo es asimetrico. Si voy a probar simetrico me toca tener guardada una llave (secreto compartido) Ksc. Asumo que ya la tienen o tambien me toca ese handshake.

Q7:

(i) Un servidor iterativo y un cliente iterativo. El cliente genera 32 consultas.
(ii) Un servidor y un cliente que implementen delegados. El número de delegados, tanto servidores como clientes, debe variar entre 4, 16 y 32 delegados concurrentes. Cada cliente genera una sola solicitud.

↳ Me toca meter threads tambien??

} Brutal