Sistema de mensajería con el DNIe

Arturo Francés y Marcos Fraile

1. RESUMEN:

En este informe se describe el desarrollo e implementación de una aplicación de mensajería con cifrado extremo a extremo, a través del DNIe, creada por nosotros. Además, se analizan las limitaciones técnicas del sistema y se realiza una valoración de la seguridad del diseño, destacando los aspectos más relevantes del proceso de desarrollo y las decisiones adoptadas.

https://github.com/marcosf22/Instant-Messaging-with-DNIe-Identity

2. FUNCIONAMIENTO DEL SISTEMA:

Para llevar a cabo este sistema de mensajería hemos descrito , 4 programas principales. El primero de ellos es el **protocol.py**. Este programa se encarga de definir cómo se construyen los paquetes que intervienen en la comunicación. Cada uno de estos paquetes se construye con una cabecera de 5 bytes. El primer byte marca el tipo de paquete que estamos enviando o recibiendo, y los otros 4 son bytes usados para el número de secuencia. Los distintos tipos son: MSG_HELLO: Para iniciar el saludo (Handshake), MSG_DATA: Mensajes de chat (cifrados), MSG_AUTH: Verificación de identidad (firma digital con DNIe), MSG_ACK: Confirmación de recepción y MSG_BYE: Aviso de desconexión. Además, estos paquetes cuentan con un payload donde se transportan las claves o mensajes cifrados en la comunicación.

El siguiente programa es el **discovery.py**. Este programa se encarga de gestionar cómo encontrar a los usuarios que están en la red. Para ello usamos mDNS o multicast DNS, el cual es un protocolo que anuncia su servicio en una red y un puerto definidos y escucha a aquellos usuarios que se están anunciando en el mismo puerto o en un puerto distinto, en la misma red. Cuando encuentra a alguien, el programa main, que explicaremos más adelante, guarda a los usuarios encontrados en una lista de contactos.

Después, tenemos el programa **crypto.py**. Este se encarga de manejar el cifrado de las claves del DNIe y del chat. Además, es el encargado de firmar los datos de los paquetes, cuando el usuario está autenticado. Para el cifrado de las conversaciones se usa un cifrado simétrico con una clave compartida temporal mediante curvas elípticas.

Y, por último, tenemos el **main.py**. Este programa llama a los anteriores para realizar la comunicación entre usuarios. En este programa se implementa la interfaz, de la que hablaremos en el siguiente apartado. También se encarga de guardar las sesiones creadas por el usuario, gestionar la persistencia de los mensajes que hay en cola, cuando uno o ambos extremos de la comunicación no están disponibles.

Una vez sabemos que hace cada programa, podemos definir el funcionamiento global de nuestro diseño. Para explicarlo, lo hemos definido en 3 fases:

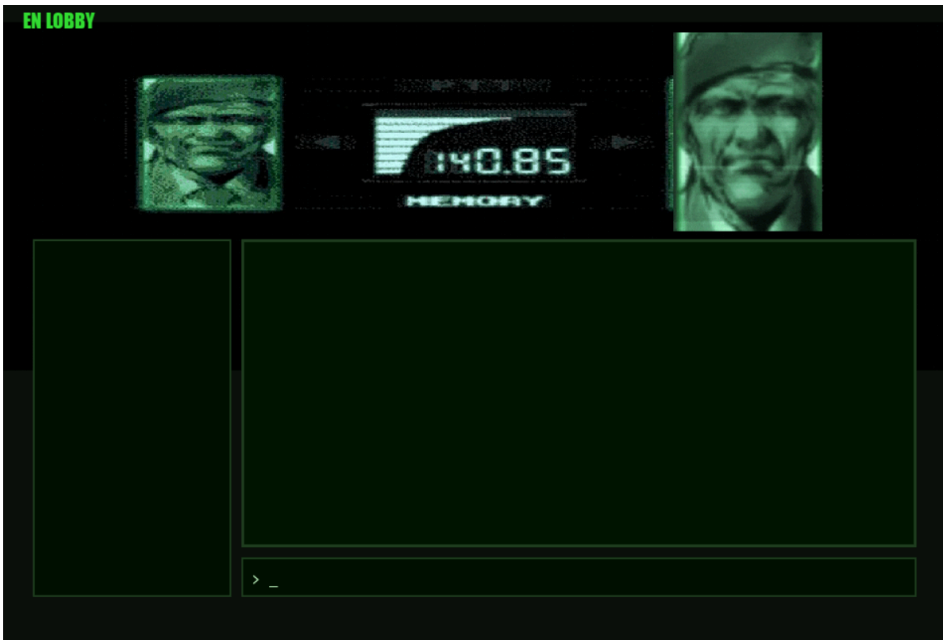
La primera fase es la de descubrimiento. En ella, nuestro sistema, nada más iniciarse, publica su servicio como hemos mencionado antes, usando el discovery.py, en cuanto haya un nuevo usuario en nuestra red, se añadirá a la lista de contactos que es visible en la interfaz.

La segunda fase, es el handshake entre usuarios. Para ello primero, se generan las claves temporales para cada usuario, luego cada usuario envía su clave pública al otro con un MSG_HELLO, y ambos calculan una nueva clave mediante su clave privada y la pública recibida. Una vez ambos tienen la misma llave de cifrado, se produce la autenticación con el DNIe. En ella, cada usuario toma la clave nueva generada y la firma con su DNIe. Esto se envia al otro extremo con un MSG_AUTH. Una vez recibido por el otro usuario, este verifica la firma y extrae el “common name” del DNI para saber el nombre de la otra persona. Si la firma es válida, aparecerá un “ok” en la interfaz.

Y, la última fase, es la del transporte de datos. En esta fase, una vez autenticados los usuarios, pueden entablar una conversación que tiene asignado un archivo JSON en el que se guardan los mensajes para que persistan, y se cifra usando la clave compartida. Cuando un usuario envía un mensaje se envía con un MSG_DATA vía UDP, y cuando el otro extremo lo recibe, este descifra, muestra el mensaje y envía un MSG_ACK (Confirmación) para que el emisor sepa que llegó (y ponga el doble tick).

3. INTERFAZ GRÁFICA:

Como nos encantan las ideas felices, esta vez hemos basado nuestra interfaz en el codec de Metal Gear Solid.



Como se ve en la imagen, a la izquierda tenemos la barra de contactos detectados en la red con mDNS, a la derecha de esta barra, tenemos la consola donde podemos escribir y ver nuestros mensajes. La gracia de la interfaz es que a cada usuario asigna una imagen de un personaje del videojuego en el que está basada la interfaz. Además, los personajes hablan cuando mandamos un mensaje. Y cuando se verifica el pin, el personaje asignado cambia, porque hemos revelado nuestra identidad.

4. TRABAJO CON LA IA Y REPARTO DE TAREAS:

A la hora de organizarnos, hemos trabajado paralelamente con 2 chats distintos de gemini cada uno de nosotros. De forma que, para desarrollar cada uno de los programas, cada uno hacía su versión y los comparamos. La decisión de separar el código en 4 programas era para tener más ordenado el código. Además, abrimos otro chat distinto para llevar a cabo la interfaz, una vez se nos ocurrió hacer

la interfaz de Metal Gear Solid. Para desarrollar la interfaz, Arturo hizo el esqueleto de la interfaz con las especificaciones que queríamos implementar y Marcos acomodó el código del main en la interfaz.