Verificadores de integridad

Contenido

Verificadores de integridad 1
Introducción 2
Descripción del software 2
Desarrollo del proyecto 3

Introducción

Con este proyecto pretendo el aseguramiento de la información con técnicas de integridad de datos en la transmisión por redes públicas como internet y evitar los diferentes tipos de posibles ataques.

Me pongo en el supuesto de una entidad financiera que busca hacer una transacción, el formato buscado es "**Cuenta origen, Cuenta destino, Cantidad transferida**"

Con este proyecto pretendo hacer frente a ataques muy ocurrentes como el de Man-in-the-middle o el de replay (en servidor y cliente).

Descripción del software

Usare Python debido a su amplio repertorio de librerías de criptografía muy útiles para lo que busco, en concreto he encontrado una librería "Cryptography.fernet", Fernet es una biblioteca de Python para la criptografía simétrica, especialmente diseñada para la facilidad de uso y seguridad en aplicaciones web y sistemas de comunicación, se adapta a la perfección al sistema que queremos diseñar. Debemos tener en cuenta que el cifrado simétrico utiliza la misma clave tanto para cifrar como para descifrar datos.

Desarrollo del proyecto

Como hemos visto antes busco un transmitir un mensaje con cuenta origen, cuenta destino y cantidad, y debo verificar la integridad en su transmisión, el nombre del algoritmo que uso para verificar la integridad y clave usada por cliente y servidor.

Para conseguir un punto más de seguridad incluyo un nonce, tanto la clave como este nonce se obtienen de la función “Ferner.generate.key()” que genera una clave segura aleatoria de 32 bytes.

Con el nonce prevengo los ataques de repetición.

La clave debe ser la misma para el cifrado y el descifrado, por lo que la establezco como variable global estática, es estática por que si la generamos automáticamente con la función antes mencionada se produciría un error ya que el cliente y el servidor generarían claves distintas.

Con Fernet creamos el objeto Fernet a partir de la clave y uso el encriptado y desencriptado que nos proporciona esta librería, Pero antes de realizar el paso de codificación decido darle una estructura al mensaje enviado (por comodidad a la hora de desencriptar), con las funciones create_msg y received_msg.

Decido crear un sistema de ficheros .txt que actúe como base de datos. Hay que aclarar que la posibilidad de que el cliente repita el nonce (autogenerado) es prácticamente imposible por eso me centro en verificar exclusivamente si el nonce recibido ha sido recibido con anterioridad y no en la comprobación de si el nonce que hemos generado ya lo habíamos generado antes. Que el nonce recibido aparezca ya en la base de datos implicaría un ataque man-in-the-middle pues el nonce debe ser único y derivará en una alerta.