CONSULTORIA 2

Equipo 4

PLACIDO FERNANDEZ CUEVAS
JORGE VILCHEZ ACOSTA
JUAN ANTONIO MORAL MARTÍNEZ

Contenido

Consulta A

A ¿Es seguro el tamaño de clave que estamos usando para la integridad de las transmisiones? Dado el siguiente mensaje y su correspondiente MAC enviados entre el cliente y la entidad bancaria, compruebe el tiempo mínimo que se tardaría en encontrar la clave, muestre el proceso a seguir y la clave obtenida, y cuál es de media el tiempo que se tarda en descubrir la clave. Razone si es su tiempo lo suficientemente alto como para garantizar la seguridad de la clave.

Para demostrar que el tamaño de la clave no es suficiente para garantizar las comunicaciones, hemos desarrollado un pequeño código en Python que se puede ejecutar en paralelo en diferentes procesadores.

Para calcular el tiempo que tardamos en descifrar la contraseña hemos utilizado la librería **time**, la paralelización la realizamos con la librería **mpi4py** y para generar la MAC usamos **hmac**.

El programa se reduce a iterar sobre los posibles valores del salt, a generar una posible MAC con esta clave y a comparar con la original.

```
def hmac_sha1(key, msg):
    return hmac.HMAC(key,msg,sha1)
msg="541158 487656 200"#Mensaje mandado
datoparcial=["","","","","","","","","","","","","","",""]
def iteraciones(i,f):
    for a in range(i,f):
        datoparcial[rank]= bytes([a])
        for b in range(0,256):
            datoparcial[rank+1] = datoparcial[rank] + bytes([b])
            for c in range(0,256):
                datoparcial[rank+2] = datoparcial[rank+1] + bytes([c])
                for d in range(0,256):
                    datoparcial[rank+3] = datoparcial[rank+2] + bytes([d])
                    h=hmac_sha1(datoparcial[rank+3],msg.encode('utf-8'))
                    if h.hexdigest() == "0a5f910eddc60e3b06f51670e83d37886804bf9a":#MAC con la que compararemos
                        print("Hola soy "+str(rank)+" y este es el resultado: "+ str(datoparcial[rank+3]))
                        print ("--Lo que tarda en encontrar la solucion---", int((time.time()-start_time)), "segundos-----")
                        break

start_time = time.time()
iteraciones(int((256/4)*rank),int((256/4)*(rank+1)))
comm.Barrier()
if rank==0:
    print ("---Lo que tarda en ejecutarse el programa completo--", int((time.time()-start_time)), "segundos-----")
    print("Este es el resultado: "+ str(datoparcial[rank+2]))
```

La media en términos generales de obtención de la clave ejecutando el programa en paralelo con un ordenador con 4 núcleos es de unos **400 segundos**. Siendo la media de ejecución del programa completo de unos **10968 segundos (182,8 minutos)**. Por lo tanto, si la clave es cambiada cada año **daría tiempo perfectamente a ser vulnerada**.

Las claves encontradas son:

Mensaje	Hash	Clave
531456 487654 200	c5173b3e13fbed7f1b41c7dfa5fd6fd6368cd3660	a6A
541157 487655 200	158413dd62eada5273a72f9fa35f4e19ddb864b8	!\xae-A
541158 487656 200	0a5f910eddc60e3b06f51670e83d37886804bf9a	$\xae S

Consulta B

B. En caso de que no considere que la clave es robusta, debería indicar el tamaño exacto de clave que sería conveniente (48 bits ?, 64 bits ? 128 bits ? …). Presente en el informe los criterios que ha considerado para llevar a cabo la selección del tamaño de clave adecuado, justificando detalladamente su elección. El cliente nos comunica que valora muy positivamente TODAS LAS PRUEBAS EMPÍRICAS QUE SE APORTEN PARA AVALAR TAL JUSTIFICACIÓN.

En nuestro caso hemos descifrado la clave en 10968 segundos, suponemos que los que realicen el ataque, tendrán una potencia de cálculo muy superior a la nuestra entre 1000 y 10000 veces la velocidad que hemos utilizado nosotros. Por tanto, tardarían del orden de 1 segundo en descifrar una clave de 4 Bytes o 32 bits.

Si utilizamos una clave de 48 bits el tiempo que utilizaría con la estimación anterior sería:

> 48 bits: 1 segundos * 256 * 256 = 65536 segundos -> 18 horas

En este caso, la clave seguiría estando comprometida, no se recomienda su uso.

Si utilizamos una clave de 64 bit el tiempo que utilizaría con la estimación de cálculo anterior sería:

> 64 bits: 1 segundo * 256 *256 *256 *256 = 136 años

En el caso de usar 64 bits, puede darse un caso poco probable de que consigan descifrar la clave, antes de que se acabe el año.

Para asegurarnos 100% de que la clave no va a ser descifrada, necesitaríamos 128 bits.

Consulta C

Desplegar un verificador de integridad en los sistemas cliente/servidor para llevar a cabo la realización de la verificación de forma práctica de los mensajes transmitidos entre un servidor y un cliente. (Valoración 35%). Se debe tener en cuenta que:

• Como entrada se recibirá el mensaje (Cuenta Origen, Cuenta Destino, Cantidad) a verificar la integridad en su transmisión, y se debe poder especificar el nombre del algoritmo que se usará para verificar la integridad, y la clave utilizada por el cliente y el servidor para cada cuenta bancaria origen.

• Output del sistema: Indicación en el cliente y servidor si se ha conservado la integridad o no se ha conservado. La salida podría ser presentada en una ventana al emisor del mensaje y en el servidor dejar constancia en un fichero de logs de los mensajes que no han llegado de forma íntegra y de la ratio de mensajes que se envían de forma íntegra/número total de mensajes enviados entre los usuarios y la entidad financiera

En esta parte hemos generado 3 claves nuevas de 64 bits para los 3 mensajes, estas claves como ya hemos podido comprobar son inviables a la hora de poder obtenerlas por la vía que hemos tomado.

Mensaje	Hash	Clave
531456 487654 200	7ba227fa483f0585b3e9fa228dcf84ef53df1942	)SG]t09{
541157 487655 200	3f340c02770e70b6e1f364cf27cc0ddcf7bc23b0	-F)jLO'q
541158 487656 200	b01fcb8abceea2c8ae3d8826edb28280db417370	'd3}kj@i

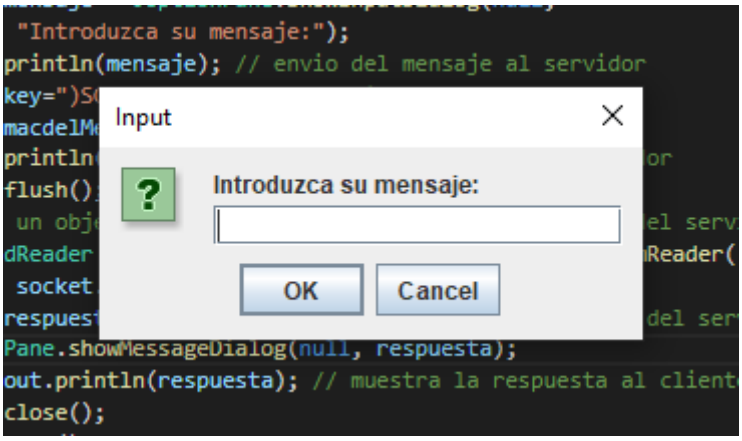
Para generar la MAC a enviar, a diferencia de en el apartado A hemos usado Java. La clase se llama **MACGen** y concretamente de ese fichero usaremos una función llamada **calculaMac(dato,key)**, esta llamará a su vez a otra función que generará la MAC teniendo como entrada el **mensaje** y el **salt** dados.

```java
public class MACGen {
    private static final String HMAC_SHA1_ALGORITHM = "HmacSHA1";

    private static String toHexString(byte[] bytes) {
        Formatter formatter = new Formatter();

        for (byte b : bytes) {
            formatter.format("%02x", b);
        }

        return formatter.toString();
    }

    public static String calculateRFC2104HMAC(String data, String key)
        throws SignatureException, NoSuchAlgorithmException, InvalidKeyException
    {
        SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(), HMAC_SHA1_ALGORITHM);
        Mac mac = Mac.getInstance(HMAC_SHA1_ALGORITHM);
        mac.init(signingKey);
        return toHexString(mac.doFinal(data.getBytes()));
    }

    public static String calculaMac(String data, String key)
            throws InvalidKeyException, SignatureException, NoSuchAlgorithmException {
        String res= calculateRFC2104HMAC(data, key);
        return res;
    }

    Run | Debug
    public static void main(String[] args) throws Exception {
        System.out.print("Esto es un test de ejecutar MACGen, es lo que devuelve la funcion calculaMac(data,key)\n");
        System.out.println(calculaMac("531456 487654 200"," a6A"));
    }

}
```

El programa funciona siguiendo una serie de pasos:

-Ejecutar el servidor, este dejará un puerto en escucha para poder enviar mensajes a este.

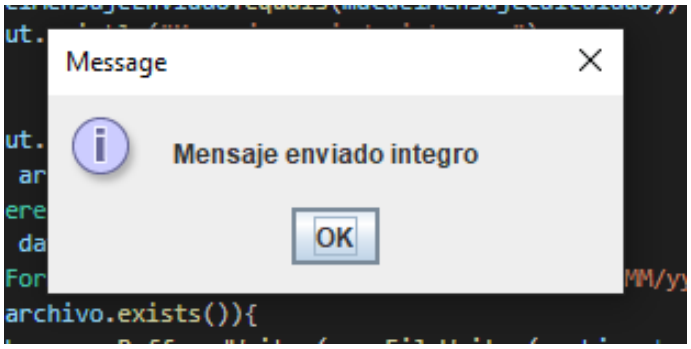
-Ejecutar el cliente, que conectará con el servidor por el puerto acordado, en este momento se le aparecerá al cliente una ventana emergente donde tendrá que escribir el mensaje que desea enviar y se enviará. El proceso tras esto es tal que: el mensaje se guarda en una variable, el programa recoge la key y el mensaje y llama a la función calculaMac para enviarla con el mensaje al servidor.

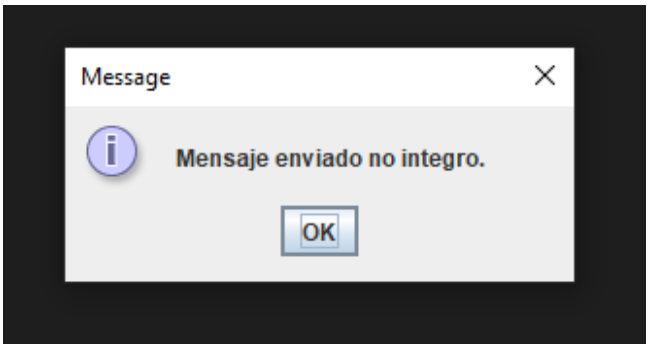



```
23          String mensaje = JOptionPane.showInputDialog(null,
24                  "Introduzca su mensaje:");
25          output.println(mensaje); // envio del mensaje al servidor
26          String key=")SG]t09{";//Key Compartida
27          String macdelMensaje = MACGen.calculaMac(mensaje, key);
28          output.println(macdelMensaje);//Envío de la MAC al servidor
29          output.flush();
```

-En este punto el servidor verificará la integridad del mensaje, generando una nueva MAC con el mensaje recibido y la key compartida, y posteriormente comparando ambos MAC para comprobar si son idénticos. Tanto si la operación ha sido vulnerada como si no lo ha sido el cliente recibirá un mensaje de vuelta.

```
42          String key= ")SG]t09{";
43          String macdelMensajeCalculado = MACGen.calculaMac(mensaje, key);//
44          System.err.println(mensaje);
45
46          if (macdelMensajeEnviado.equals(macdelMensajeCalculado)) {
47              output.println("Mensaje enviado integro ");
48              a++;
49          } else {
50              output.println("Mensaje enviado no integro.");
51              File archivo= new File("./logf.txt");
52              BufferedWriter bw;
53              Date date =new Date();
54              DateFormat h= new SimpleDateFormat("HH:mm:ss dd/MM/yyyy");
55              if (archivo.exists()){
56                  bw= new BufferedWriter(new FileWriter(archivo,true));
57              }else{
58                  bw= new BufferedWriter(new FileWriter(archivo,false));
59              }
60              f++;
61              bw.write("["+h.format(date)+"]"+"Mensaje enviado no integro---- El ratio de fallo es de: "+(f/f+a)*100+"%\n")
62              bw.close();
63
64          }
```





-Si la integridad del mensaje se ha visto comprometida y este ha sido modificado entonces se generará un registro en un log, esta línea contendrá la fecha del acontecimiento, el error dado y el porcentaje de fallos contra los de acierto.

logf.txt

```
1  [00:16:06 31/10/2020]Mensaje enviado no integro---- El ratio de fallo es de: 100%
2  [00:16:20 31/10/2020]Mensaje enviado no integro---- El ratio de fallo es de: 100%
3  [11:28:18 02/11/2020]Mensaje enviado no integro---- El ratio de fallo es de: 100%
4  [11:28:48 02/11/2020]Mensaje enviado no integro---- El ratio de fallo es de: 100%
5  [11:31:01 02/11/2020]Mensaje enviado no integro---- El ratio de fallo es de: 100%
6  [11:31:38 02/11/2020]Mensaje enviado no integro---- El ratio de fallo es de: 100%
7  [11:31:59 02/11/2020]Mensaje enviado no integro---- El ratio de fallo es de: 100%
8
```

Consulta D

Explique las medidas de seguridad tomadas para evitar problemas de incumplimiento de la Política de Seguridad relacionada con la integridad de las transmisiones usando el algoritmo de MAC. Por ejemplo, para evitar posibles ataques de man-in-the-middle, replay o similares. (Valoración 15%)

Los ataques que podríamos tener al usar esta tecnología como man-in-the-middle o replay afectan a los mensajes tomados por el receptor, pudiendo ser fraudulentos, la solución y las medidas tomadas para evitar este problema de integridad es utilizar el algoritmo MAC y adjuntar un timestamp en la cabecera o en el propio mensaje. Con MAC, como ya hemos podido comprobar antes, usamos una clave que solo conocen el cliente y el servidor, y un algoritmo para generar un hash de la unión de la clave+mensaje. Para asegurar que la clave sea más difícil de corromper hemos usado una de 64 bits, la que nos daría, si se recorriese por completo, un tiempo de algo más de 100 años, completando el requisito de la longitud de la clave, la cual es cambiada cada año por lo que si es viable su obtención en menos de un año entonces podemos decir que es vulnerable. El objetivo del uso del algoritmo MAC en todo esto es que, si existe algún tipo de ataque como los comentados antes, al llegar el mensaje cambiado al receptor, este cuando compruebe el hash con la clave podría percatarse de que el mensaje ha sido cambiado o de que incluso el mensaje no concuerda con el hash enviado, en ambos casos el mensaje se remitiría a un log de errores. Con respecto a los ataques de replay podríamos implementarlo añadiendo un timestamp a la cabecera del mensaje o en el propio mensaje, si vemos que el tiempo en esa marca es idéntico a otra con el mismo mensaje entonces estaríamos sufriendo un ataque de replay.

Consulta E

Se debe detallar el procedimiento a llevar a cabo para que el cliente y servidor tengan la misma clave para hacer la comprobación de la integridad. (Valoración 15%). Proponer una política adecuada para la entrega de las claves a los clientes, informando del procedimiento que se deba seguir, las personas y/o sistemas implicados, y la periodicidad. Detalle como el banco debe custodiar las claves hasta que sean entregadas, y en que soporte se le entregarán. Tenga en cuenta la información que el banco conoce de sus clientes para asegurar que el proceso sea lo más seguro posible.

El problema que se nos plantea es hacer llegar la clave de forma totalmente segura a cualquiera de nuestros clientes. Para ello, el método más efectivo y seguro es usar un sistema de cifrado para enviar esa clave. Podemos basarnos en un sistema de cifrado asimétrico, ya que aunque sean más lentos que los sistemas de cifrado simétrico, sólo tenemos que transmitir un mensaje, por lo que en este aspecto no nos supondrá un problema.

El sistema que se propone es RSA, el cual consiste en una clave pública para encriptar el mensaje y una clave privada para desencriptarlo. La clave pública es conocida por todos los usuarios mientras que la clave privada solamente es conocida por quien va a desencriptar el mensaje.

Para transmitir la clave sería muy sencillo, primero determinaríamos la clave pública del cliente, la cual no importa si es interceptada ya que solo puede usarse para encriptar. Una vez hemos determinado su clave pública, encriptamos la clave a enviar y la enviamos. No importa si otra persona recibe esta clave encriptada ya que sólo podrá desencriptarla el cliente haciendo uso de su clave privada. Una vez el cliente la desencripte con su clave privada, poseerá la clave de una forma totalmente segura y por medio de internet, lo cual le permite estar en cualquier parte.

Para usar este método de cifrado asimétrico, se deberá disponer de un certificado digital, por lo que es un método perfecto para las empresas y las personas que dispongan del mismo, pero debemos pensar también en el cliente medio, el cual no suele disponer de este tipo de certificado. Para este tipo de clientes medios, podemos realizar la transferencia después de una verificación en dos pasos, como se suele hacer actualmente, la cual consistiría, por ejemplo, en enviar un código por mensaje al número de teléfono del cliente y posteriormente, utilizar dicho código en alguna página web para confirmar su identidad.

En cuanto a la periodicidad, si las claves deben renovarse cada año, este mismo proceso deberá repetirse cada año. Esto dependerá del tamaño de la clave que se haya seleccionado anteriormente.

Para almacenar las claves MAC, lo más recomendable sería almacenarlas ya cifradas usando un cifrado simétrico, puesto que las claves se almacenarían en algún servidor del propio banco, al cual se le puede dar la clave simétrica necesaria. Esta parte es muy importante ya que puede poner en riesgo la seguridad del banco, deben almacenarse en un servidor seguro al que sea muy difícil acceder.