Antonio Jiménez Sevilla

PSP07

Declaramos las variables para crear la clave SecretKeySpec, la contraseña, el fichero y el contenido del fichero

```
public static void main(String[] args) {
    FileWriter fw = null;
    try {
        // declaro las variables
        SecretKeySpec clave = null;
        String pwd = "1234";//passwrod
        String ruta = "fichero";//nombre del fichero
        //lo que se va a escribir en el fichero
        String textoFichero = "Tarea de PSP07 super secreta";
        //objeto para declarar la ruta del fichero y crear un fichero
        //para leerlo despues
```

Crear un objeto para crear un fichero, y obtenemos el hash del password 1234, para comprobar que el hash es de 256 bits o 32 Bytes, uso un array de Bytes e imprimo por pantalla la longitud el array para comprobar que son 32 Bytes, *"hashed.lenght"*. De esos 32 Bytes me quiero quedar con los 24 primeros, utilizo la clase Arrays.copyOf .

```
//para leerlo despues
File fichero = new File(ruta);
//creo el fichero
fichero.createNewFile();
fw = new FileWriter(fichero);
BufferedWriter bw = new BufferedWriter(fw);//se almacena en el buffer
bw.write(textoFichero);//se escribre el contenido
bw.close();
MessageDigest sha256;//clase Message direct para crear contraseña HASh
try {
    sha256 = MessageDigest.getInstance("SHA-256");//se instancia
    byte[] hashed = sha256.digest(pwd.getBytes("UTF-8"));//se codifica
    //miro los bytes que tiene hashed y tiene 32Bytes
    // System.out.println("---"+hashed.length);
    byte[] hashed192 = Arrays.copyOf(hashed, 24);//24 bytes son 192 bits
```

Para cifrar el fichero creamos un método estático *cifrarFichero* este encripta el fichero que se pasa como parámetro devuelve el valor de la clave privada utilizada en encriptación. El fichero encriptado lo deja en el archivo de nombre *fichero.cifrado* en el mismo directorio.
Una vez obtenido el array de bytes, construyo una clave AES con la clase SecretKeySpec.
Usamos la clase Cipher con la que creamos un objeto y encriptamos el mediante el algoritmo *AES/ECB/PKCS5Padding.* Iniciamos el algoritmo y le pasamos la clave SecretKeySpec.

```
private static SecretKeySpec cifrarFichero(String file, byte[] pass)
        throws NoSuchAlgorithmException, NoSuchPaddingException,
        FileNotFoundException, IOException, IllegalBlockSizeException,
        BadPaddingException, InvalidKeyException {

    FileInputStream fe = null; //fichero de entrada
    FileOutputStream fs = null; //fichero de salida
    int bytesLeidos;

    //1. Crear e inicializar clave
    System.out.println("1.-Genera clave AES");
    //crea un objeto para generar la clave usando algoritmo AES
    SecretKeySpec skeySpec = new SecretKeySpec(pass, "AES");
    mostrarBytes(skeySpec.getEncoded()); //muestra la clave
    System.out.println();

    //Se Crea el objeto Cipher para cifrar, utilizando el algoritmo AES/ECB/PKCS5Padding
    Cipher cifrador = Cipher.getInstance("AES/ECB/PKCS5Padding");
    //Se inicializa el cifrador en modo CIFRADO o ENCRIPTACIÓN
    cifrador.init(Cipher.ENCRYPT_MODE, skeySpec);
    //se cifra el fichero
    System.out.println("2.-Fichero cifrado ..... ");
    //declaración  de objetos
```

Mediante el método mostrarBytes podemos mostrar la clave por pantalla.

```
private static void mostrarBytes(byte[] buffer) {
    System.out.write(buffer, 0, buffer.length);
}
```

Ciframos el fichero, se lee de 1000 en 1000, se le pasa los fragmentos leídos al cifrador. Luego mientras no se al final del fichero que es -1, pasa texto al cifrador y lo cifra, asignándolo a *bufferCifrado.* Se completa el cifrado y se graba el final del texto cifrado si hay alguno.

```
    System.out.println("2.-Fichero cifrado ..... ");
    //declaración  de objetos
    byte[] bufferCifrado;
    byte[] leidos = new byte[1000];//leemos de 1k en 1k
    fe = new FileInputStream(new File(file)); //objeto fichero de entrada
    fs = new FileOutputStream(file + ".cifrado"); //fichero de salida
    //lee el fichero de 1k en 1k y pasa los fragmentos leidos al cifrador
    bytesLeidos = fe.read(leidos, 0, 1000);
    while (bytesLeidos != -1) {//mientras no se llegue al final del fichero
        //pasa texto claro al cifrador y lo cifra, asignándolo a bufferCifrado
        bufferCifrado = cifrador.update(leidos, 0, bytesLeidos);
        fs.write(bufferCifrado); //Graba el texto cifrado en fichero
        bytesLeidos = fe.read(leidos, 0, 1000);
    }
    bufferCifrado = cifrador.doFinal(); //Completa el cifrado
    fs.write(bufferCifrado); //Graba el final del texto cifrado, si lo hay

    //Cierra ficheros
    fe.close();
    fs.close();
    return (SecretKeySpec) skeySpec;
}
/Aqui tiene que leer el fichero y sacarlo por pantalla
```

Este método desencripta el fichero pasándole como parámetro la clave privada SecretKeySpec que necesita para desencriptar, key.

```
private static void descifrarFichero(SecretKeySpec key)
        throws NoSuchAlgorithmException, NoSuchPaddingException,
        FileNotFoundException, IOException, IllegalBlockSizeException,
        BadPaddingException, InvalidKeyException {
```

En este método desciframos, y leemos el fichero por pantalla, esta vez en la clase Cipher, al crear el objeto cifrador e inicializarlo, en vez de encriptar, tendremos que desencriptar, con le que usaremos DECRYPT_MODE. Y le pasaremos la clave SecretKeySpec para poder leer el contenido encriptado anteriormente utilizando la misma clave.

```
    private static void descifrarFichero(SecretKeySpec key)
            throws NoSuchAlgorithmException, NoSuchPaddingException,
            FileNotFoundException, IOException, IllegalBlockSizeException,
            BadPaddingException, InvalidKeyException {
        FileInputStream fe = null; //fichero de entrada
        File file = new File("fichero.cifrado");
        byte[] arrayDescif = null;//para leer lo descifrado
        fe = new FileInputStream(file);
        Cipher desCifrador = Cipher.getInstance("AES/ECB/PKCS5Padding");
//      Poner cifrador en modo DESCIFRADO o DESENCRIPTACIÓN
        desCifrador.init(Cipher.DECRYPT_MODE, key);
```

Al igual que en el método anterior leemos el fichero de 1k en 1k, se pasa texto cifrado al cifrador y lo descifra, asignándolo a buffer. Creamos un String para poder sacar por pantalla la lectura del fichero cifrado. Usamos el array de bytes para descifrar y leer y el objeto de la clase Cipher desCifrador para poder leer descifrar el texto

```
    int bytesLeidos;
    byte[] buffer = new byte[1000]; //array de bytes
    //lee el fichero de 1k en 1k y pasa los fragmentos leidos al cifrador
    bytesLeidos = fe.read(buffer, 0, 1000);
        //pasa texto cifrado al cifrador y lo descifra, asignándolo a buffer
        arrayDescif = desCifrador.update(buffer, 0, bytesLeidos);
        bytesLeidos = fe.read(buffer, 0, 1000);
        //Utilizo este String para poder sacar por pantalla el texto descifrado
        //mas el descifrado
   String descifrar = new String(arrayDescif) + new String(desCifrador.doFinal());
    System.out.println("3.-Leemos el fichero y sacamos por pantalla :\n" + descifrar)
    //cierra archivos
    fe.close();

}
```

Para acabar en la clase principal, main, Utilizamos los métodos creados. Llamamos al método que encripta el fichero que se pasa como parámetro y el array de 192 bits. Al método descifrar fichero le pasamos la clave.