**Laboratorio 2**

**Criptografía y**

**Seguridad Informática**

Jorge Rodríguez Fraile, 100405951

# Bouncy Castle

## Crear archivo de texto

Texto:

DEFINIR AUNQUE SEA BREVEMENTE QUE HEMOS HECHO EN LOS PROCEDIMIENTOS, HASH Y CIFRADO. HAY QUE INDICAR CUANTO OCUPA EN LA SALIDA FINAL CADA SECCION DE LO CIFRADO, CLAVE K Y ID, EL RESTO ES EL MENSAJE

## **Sección** **DES**

Cuestiones

1. Consulte la siguiente página web y evalúe con la información proporcionada si hubiese sido más adecuado utilizar la clase java.util.Random de Java. <https://wiki.sei.cmu.edu/confluence/display/java/MSC02-J.+Generate+strong+random+numbers>

The Java API provides a PRNG, the java.util.Random class. This PRNG is portable and repeatable. Consequently, two instances of the java.util.Random class that are created using the same seed will generate identical sequences of numbers in all Java implementations. Seed values are often reused on application initialization or after every system reboot.

1. Observe el método “generateKey” de la clase “DES.java”. ¿Por qué se multiplica la longitud preestablecida de una clave DES por 8? Cambie dicha longitud por otra, ¿qué ocurre durante la ejecución?

Multiplica por 8 para que tenga una longitud de 64 bits.

Probamos multiplicando por 4, y da el error: DES key must be 64 bits long.

Cuestiones

1. Consulte la documentación de Bouncy Castle y cambie el modo de Bloque del cifrador al modo CFB y OFB. Genere un archivo cifrado para cada modo de bloque a partir de un mismo texto en claro. Si cambiamos algún byte del archivo cifrado (es decir, introducimos errores), ¿qué ocurre al intentar descifrarlo en cada uno de los modos?

CFB

Buena

9cd3914cd04c8c653f89e14a0613de0afcf72dbbcedb7a9ae052463de750e7add93ca449fa3ebc640deb2c1ed204ebc3f2d4137c0993afd018f698a0b585696806d5fdb5beeb72df14851fe8123bd9f9af7537417101c0627cea8f17c3c14268634a058984d363a92817d754597765fda0fdd5d89bb1ce93d09289e067a403d644b2fec059ab72e596fcd0e7a2d2991b074d184c70e50ec785d80e6d28f16ac5c113544122818e4565c95d499e6b9ec85c527a558f6ee18ff4bb441eccb7ee2ecdcea2c5b4510a17

Descifra correctamente.

Modificado:

9cd3914cd04c8c653f89e14a0613de0afcf72dbbcedb7a9ae052463de750e7add93ca449fa3ebc640deb2c1ed204ebc3f2d4137c0993afd018f698a0b585696806d5fdb5beeb72df14851fe8123bd9f9af7517417101c0627cea8f17c3c14268634a058984d363a92817d754597765fda0fdd5d89bb1ce93d09289e067a403d644b2fec059ab72e596fcd0e7a2d2991b074d184c70e50ec785d80e6d28f16ac5c113544122818e4565c95d499e6b9ec85c527a558f6ee18ff4bb441eccb7ee2ecdcea2c5b4510a17

El error se propaga en un bloque completo.

OFB

Buena:

9cd3914cd04c8c65ddddedcaf20da1060c2cc48d57595e75393b36760c71f893fef80eb99cf46d630c42f6283689db5d7859dffe6fe21b43ddaa2fa5a744c640166984ef653bcd3917dc65c0b408eec623981c2db929d2ac5c75d3893c3431c9f4ddd5cb98773a92decd7e8e91b981a479ba4c20d176ea902aedfadc024280947bc2b0c67ee482b6e2d175008229cdbea1ade7963943eaef53b96459fb4c5993285735444722eab3c71ed06680de37669f2160b57e4e5782bc0cb18aa6d7f5b28a8735b529984ad1

Descifra correctamente.

Modificado:

9cd3914cd04c8c65ddddedcaf20da1060c2cc48d57595e75f93b36760c71f893fef80eb99cf46d630c42f6283689db5d7859dffe6fe21b43ddaa2fa5a744c640166984ef653bcd3917dc65c0b408eec623981c2db929d2ac5c75d3893c3431c9f4ddd5cb98773a92decd7e8e91b981a479ba4c20d176ea902aedfadc024280947bc2b0c67ee482b6e2d175008229cdbea1ade7963943eaef53b96459fb4c5993285735444722eab3c71ed06680de37669f2160b57e4e5782bc0cb18aa6d7f5b28a8735b529984ad1

El error solo afecta a un carácter.

## Sección AES

Cuestiones

1. Observe el método “generateKey” de la clase “AES.java”. ¿Por qué se añade blocksize a la longitud de la clave AES en la generación del número aleatorio? Cambie dicha longitud por otra, ¿qué ocurre durante la ejecución?

Primero se cogen los primeros 24 bytes (0-23) que son la clave y los siguientes representan el Vector Inicial que son 16 bytes (24-39), como ha generado la semilla con la longitud conjunta debemos coger lo que le corresponde a cada uno. De ahí que se use blocksize.

No parece afectar el cambio.

1. Observe los métodos encrypt y decrypt de la clase AES.java. ¿pueden unificarse ambos métodos en uno sin modificar el resultado de la ejecución?

Podría ponerse para que el encrypted entre directamente al decrypter, de tal manera que podamos dar de salida ambos valores, el encriptado y el resultado de desencriptar que sería la entrada.

1. En la librería BouncyCastle, el algoritmo Rijndael es un modo “rápido” de cifrado AES que permite distintos tamaños de clave mientras que ésta sea múltiplo de 32 bits. Modifique el método encrypt para que se cifre mediante el algoritmo Rijndael que sea compatible con la función decrypt ya existente. La implementación de Rijndael en BouncyCastle se realiza en la clase RijndaelEngine

Hecho, el archivo RijndaelCypher.encaes.

## **Sección Funciones** **Resumen**

Cuestiones

1. MD5 y SHA1 son funciones hash rotas hoy en día, por lo que no deberían usarse. Consulte la documentación de BouncyCastle e implemente una nueva opción que genere resúmenes usando SHA512.

Hash de SHA512:

a742c4b5d51a18f8327bd7cbb3fe044037d909a682783c97b9a1ebdcfc5dcd8121c254c8b1fe8cf59dc557e97f94e5adea53af604f1259d156e5059f96af94a1

## Generación de par de claves RSA

Observe el método “generateKey” de la clase “RSA.java”. En este caso, y para que el alumno esté familiarizado con el formato, la información de las claves se almacena en archivos en Base64, no en Hexadecimal, usando la clase Base64Encoder

El formato Base64 es un formato de representación de bytes en forma de caracteres ASCII que permite intercambiar información entre distintos ordenadores, sistemas operativos sin que la representación de la información dependa del modo de almacenamiento de ésta entre origen destino (Big-Endian / Little-Endian, etc.). Este formato es especialmente útil cuando se desea transmitir información entre, por ejemplo, dispositivos móviles y servidores Unix en internet. Adicionalmente, esta forma de representación es la estándar para el intercambio de Certificados y Claves públicas y privadas en criptografía

## Cifrado/Descifrado RSA

Cuestiones

1. Realice el cifrado de dos ficheros de texto, uno de ellos con una longitud menor en bytes que la longitud de clave usada y otro mayor. ¿Qué ocurre cuando se intenta cifrar un texto mayor que la longitud de la clave?

Da error cuando se trata de cifrar un texto cuando es más largo que la clave, sin embargo, no hay problema porque se a menor que la clave.

1. Pruebe a generar, cifrar y descifrar texto con claves de distinto tamaño (por ejemplo, 256, 1024 y 2500) ¿qué ocurre?

Si es menor que 512 la longitud la clave da error.