

TEMA 5: TÉCNICAS DE PROGRAMACIÓN SEGURA

Programación de Servicios y Procesos

José Manuel García Sánchez



ÍNDICE

- ▶ Criptografía
- ▶ Características de los servicios de seguridad
- ▶ Estructura de un sistema secreto
- ▶ Modelo de clave privada
 - Función hash
- ▶ Modelo de clave pública
 - Firma digital
- ▶ Servicios en red seguros
- ▶ Control de acceso



Criptografía

CRIPTOGRAFÍA

Arte de escribir con clave secreta o de un modo enigmático

Ciencia que trata de conservar los secretos o hasta el arte de enviar mensajes en clave secreta aplicándose a todo tipo de información, tanto escrita como digital, la cual se puede almacenar en un ordenador o enviar a través de la red.

ENCRIPTAR

Acción de proteger la información mediante su modificación utilizando una clave.



Criptografía

- ▶ Aplicaciones de la Criptografía:
 - **Identificación y autenticación.** Identificar a un individuo o validar el acceso a un servidor.
 - **Certificación.** Esquema mediante el cual agentes fiables validan la identidad de agentes desconocidos (como usuarios reales).
 - **Seguridad de las comunicaciones.** Permite establecer canales seguros para aplicaciones que operan sobre redes que no son seguras.
 - **Comercio electrónico.** El empleo de canales seguros y mecanismos de identificación posibilita permite que las empresas y los usuarios tengan garantías de que las operaciones no van a ser espiadas ni modificadas, reduciéndose el riesgo de fraudes.



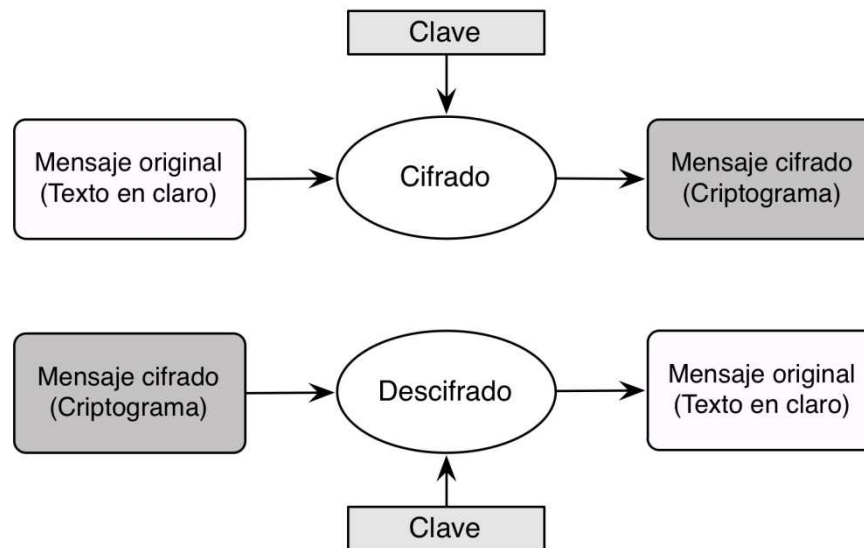
Características de los servicios de seguridad

- ▶ **Confidencialidad:** se trata de asegurar que la comunicación solo pueda ser vista por los usuarios autorizados, evitando que ningún otro pueda leer el mensaje. Suele ir acompañada de la autenticación de los usuarios que participan en la misma.
- ▶ **Integridad** de la información: se trata de asegurar que el mensaje no haya sido modificado de ningún modo por terceras personas durante su transmisión.
- ▶ **Autenticación:** se trata de asegurar el origen, autoría y propiedad de la información de quien envía el mensaje.
- ▶ **No repudio:** se trata de evitar que la persona que envía el mensaje o realiza una acción niegue haberlo hecho ante terceros. Al igual que la característica de confidencialidad, necesita de la autenticación del origen de la información.



Estructura de un sistema secreto

- ▶ Un sistema secreto actual se encuentra definido por dos funciones:
 - Función de **cifrado**.
 - Función de **descifrado**.
- ▶ La **clave** es el parámetro que especifica una transformación concreta dentro de todas las posibles sustituciones que se podrían realizar con la función de cifrado.




Ejemplo de cifrado de un mensaje

- ▶ cifra el mensaje “Hola, QuÉ tal” con un desplazamiento de 6 caracteres,
 - ¿cuál es el texto cifrado?
 - ¿y si lo desplazamos 27?
 - ¿cómo podríamos atacar un sistema de **cifrado tipo César**?



Cifrado de un mensaje. **Solución**

- ▶ Para la resolución de la primera pregunta solamente debemos sustituir cada letra por la letra correspondiente al número de orden en el abecedario resultado de sumar al orden de la letra a sustituir el número 6 y hacerle el módulo 27. Así, por ejemplo, a la letra H (orden 8 en el abecedario) le corresponde la letra de orden $(8 + 6) \bmod 27 = 14$ (N).
 - ▶ Haciendo esto para todas las letras, el mensaje cifrado quedaría: "NUQG, WAK ZGQ"
 - ▶ En el caso de un desplazamiento de 27, como hay 27 letras en el castellano, al realizar la operación la letra siempre resulta ser la misma: $(X + 27) \bmod 27 = X$. En este sentido, el mensaje al desplazarse quedaría tal y como es el texto en claro: HOLA, QUÉ TAL
 - ▶ Para descifrar un mensaje encriptado con el cifrado César, basta con probar todos los posibles números (desde 1 hasta el número máximo de letras [27 en castellano]) hasta conseguir el mensaje que tenga sentido.
- 

Criterios de Shannon

- ▶ Todos los cifrados se deben construir en base a:
 - **Confusión:** distribuir las propiedades estadísticas (redundancia) de todos los elementos del mensaje sobre el texto cifrado. Ej [Alterar posición de caracteres](#)
 - **Difusión:** dificultar la relación entre la clave y el texto cifrado mediante algoritmos para dificultar su descifrado.
- ▶ Características deseables que se deben cumplir en las funciones:
 1. El análisis estadístico del texto cifrado para descifrarlo debe suponer tal cantidad de trabajo que no sea rentable hacerlo por el envejecimiento de la propia información contenida en él.
 2. Las claves deben ser de fácil construcción y sencillas.
 3. Los sistemas secretos, una vez conocida la clave, deben ser simples pero han de destruir la estructura del mensaje en claro para dificultar su análisis.
 4. Los errores de transmisión no deben originar ambigüedades.
 5. La longitud del texto cifrado no debe ser mayor que la del texto en claro.



Criterios de Shannon

- ▶ En este sentido, la seguridad depende tanto del algoritmo de cifrado como del nivel de secreto que se le dé a la clave. Si se pierde una clave, o es fácilmente averiguable (dependiente de los datos del poseedor de la clave: fecha de nacimiento, palabra relacionada, nombre relacionado, o hasta conjunción de todo ello) se pone en peligro todo el sistema. Saber elegir una clave y establecer métodos para cuidarla es un requisito muy importante para proporcionar un sistema seguro.
- ▶ Además del nivel de seguridad de la clave, existen diferentes tipos de claves, cada una con sus ventajas e inconvenientes:
- ▶ Claves **simétricas** (K_s): las claves de cifrado y descifrado son la misma. El problema que se plantea con su utilización es cómo transmitir la clave para que el emisor (que cifra la información) y el receptor de la información (descifra) tengan ambos la misma clave. Dan lugar a lo que se denomina **modelo de clave privada**.
- ▶ Claves **asimétricas** (K_p y K_c): las claves de cifrado y descifrado son diferentes y están relacionadas entre sí de algún modo. Dan lugar al **modelo de clave pública**.

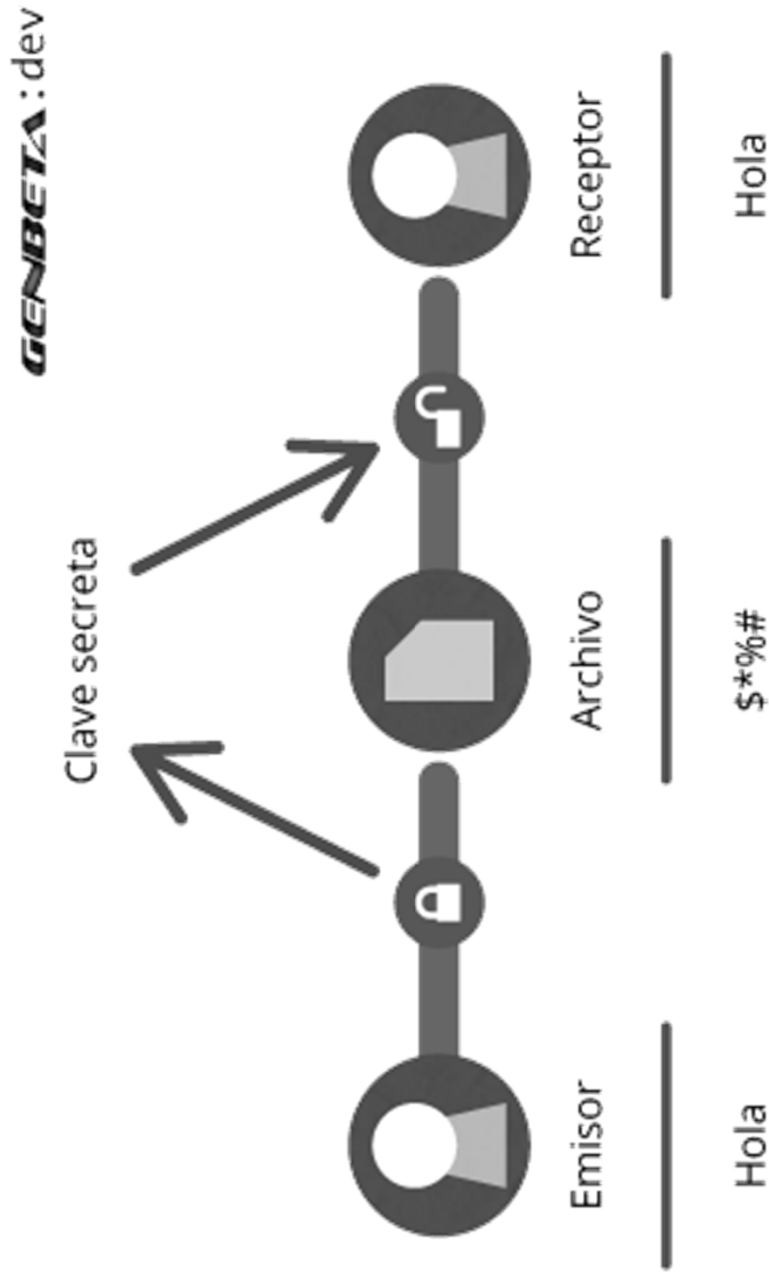


Tipos de claves

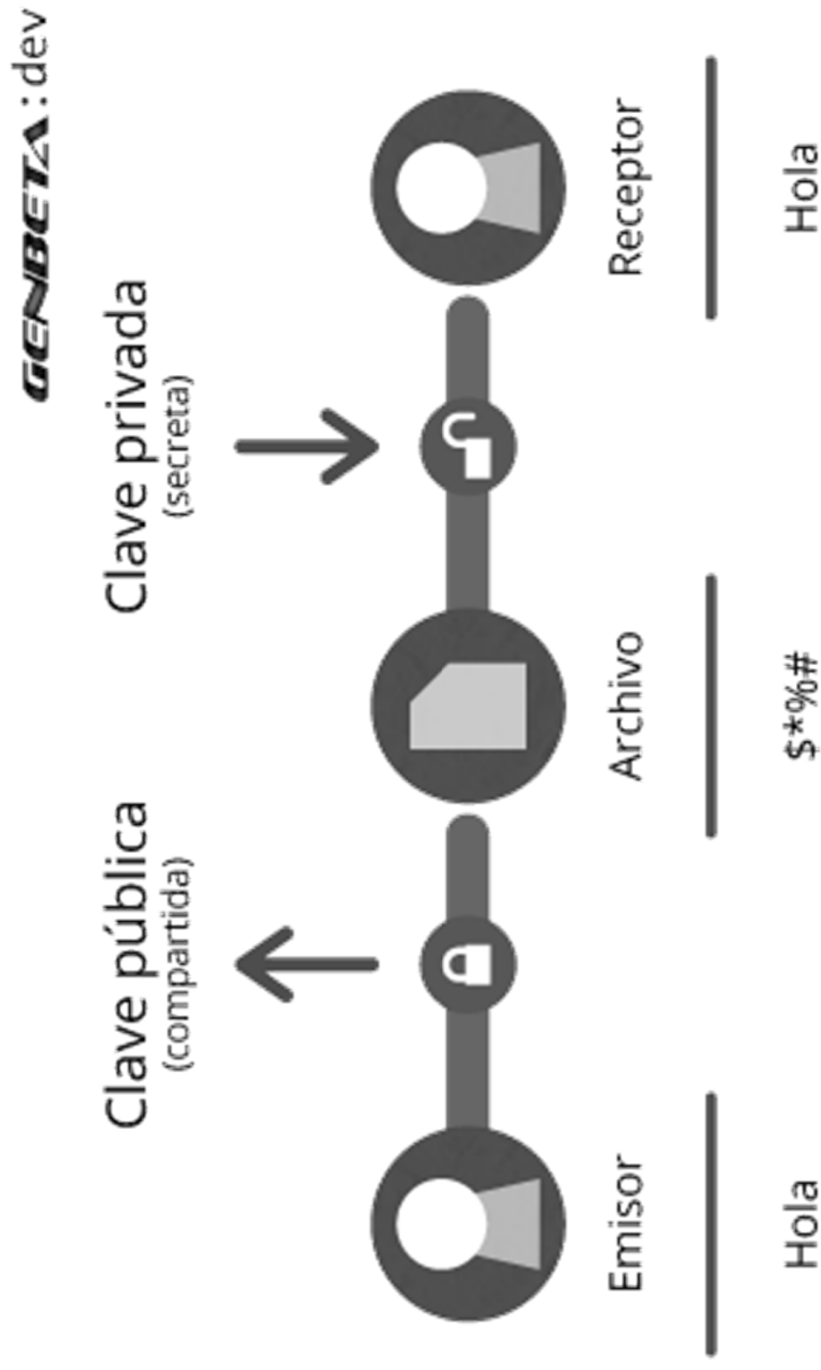
- ▶ La seguridad depende tanto del algoritmo de cifrado como del nivel de secreto que se le dé a la clave.
- ▶ Tipos de claves:
 - Claves **simétricas** (K_s)
 - Las claves de cifrado y descifrado son la misma.
 - El problema es cómo transmitir la clave para que el emisor (que cifra la información) y el receptor de la información (descifra) tengan ambos la misma clave.
 - Dan lugar a lo que se denomina **modelo de clave privada**.
 - Claves **asimétricas** (K_p y K_c)
 - Las claves de cifrado y descifrado son diferentes y están relacionadas entre sí de algún modo.
 - Dan lugar al **modelo de clave pública**.



Criptografía Simétrica



Criptografía asimétrica



Criptografía híbrida

- ▶ Es un método criptográfico que usa tanto un cifrado simétrico como un asimétrico.
- ▶ Emplea el cifrado de clave pública para compartir una clave para el cifrado simétrico.
- ▶ El mensaje que se esté enviando en el momento, se cifra usando la clave y enviándolo al destinatario.
- ▶ Ya que compartir una clave simétrica no es seguro, la clave usada es diferente para cada sesión.



Modelo de clave privada

- ▶ Tanto el emisor como el receptor del mensaje utilizan la misma clave K_s .
- ▶ Presenta un buen rendimiento
- ▶ Problemas:
 - Cómo el emisor y el receptor obtienen la misma clave, Necesidad de emplear claves distintas para comunicarse con cada uno de los posibles receptores de la información.
 - La validez de una clave se pone en entredicho a medida que se va utilizando. Cuantos más mensajes se cifren con la misma clave, más expuesta estará a un análisis estadístico.



Algoritmos de cifrado simétrico

▶ DES

Estándar por el Gobierno de los EEUU para comunicaciones desde 1976
Cifra un texto de una longitud fija en otro texto cifrado de la misma longitud.

Tres fases:

1. Permutación inicial: para dotar de confusión y difusión al algoritmo.
2. Sustitución mediante cajas-S previamente definidas que comprimen la información, la permutan y la sustituyen.
3. Permutación final inversa a la inicial.

Clave de 56 bits para modificar las transformaciones realizadas..

▶ AES

Sustituto del DES

Estándar de cifrado por el Gobierno de EEUU desde el año 2000.

Su desarrollo se llevó a cabo de forma pública y abierta

Sistema de cifrado por bloques

Maneja longitudes de clave y de bloque variables, entre 128 y 256 bits.

Rápido y fácil de implementar.



Programación de cifrado

- ▶ El lenguaje Java proporciona herramientas para el manejo de claves y mecanismos de cifrado.
- ▶ Paquetes Java
 - `import java.security.*`
 - `import javax.crypto.*`
- ▶ Los componentes básicos para el cifrado en Java son:
 - La interfaz **Key**. (clase `secretKey`)
 - La Interfaz **KeySpec**. (clase `DESKeySpec`)
 - La clase **Cipher**.



Generadores y factorías de claves

- ▶ Generadores de claves: Se utilizan para crear objetos de clases que implementan la interfaz *Key*, es decir, para crear claves nuevas. Clave Opaca
 - Ej. KeyGenerator (javax.crypto.KeyGenerator)
- ▶ Factorías de claves (SecretKeyFactory): Se emplean para obtener versiones transparentes (*KeySpec*) a partir de de claves opacas (*Key*) y viceversa.
- ▶ Sus clases suelen disponer de un método llamado getInstance() que sirve para crear instancias de estas clases



Interfaz *Key*

- ▶ **Representa una clave**, que se puede emplear para operaciones de cifrado y descifrado.
- ▶ En Java toda clase que represente una clave debe implementar esta interfaz y tener **tres partes fundamentales**:
 - El algoritmo: El nombre de la función de cifrado y descifrado.
 - La forma codificada: Es una representación de la clave.
 - El formato: Es la forma en la que se encuentra codificada la clave.



Interfaz Key. Clases Derivadas

Clase que implementa: extends [Serializable](#)

La implementan las siguientes clases:

[PrivateKey](#), [PublicKey](#), [RSAPrivateKey](#), [RSAPublicKey](#), [SecretKey](#)

Metodos	
Tipo retorno	Metodo y descripción
<u>String</u>	<u>getAlgorithm()</u> Devuelve nombre del algoritmo standard de la clave.
byte[]	<u>getEncoded()</u> Devuelve la clave en su formato de codificación primaria, o nulo si esta clave no es compatible con la codificación.
<u>String</u>	<u>getFormat()</u> Devuelve el nombre del formato de codificación principal de esta clave o nulo si esta clave no admite la codificación.



Interfaz *KeySpec*

- ▶ Las clases que implementan la **interfaz Key** almacenan las claves de forma **opaca**: No se proporciona acceso a sus componentes internos.
- ▶ Por el contrario, la interfaz ***KeySpec***, sirve para representar claves en formato **transparente** (sus componentes internos son accesibles), lo que se emplea para poder distribuir las.



Interfaz KeySpec. Clases Derivadas

La implementan las siguientes clases:

DESKeySpec, RSAPrivateKeySpec, RSAPublicKeySpec, SecretKeySpec,

public class **DESKeySpec** extends Object implements KeySpec

Esta clase especifica una clave DES

Constructores

Constructor y Descripción

DESKeySpec(byte[] key)

Crea un objeto DESKeySpec utilizando los primeros 8 bytes de la clave, como material para la clave DES.

DESKeySpec(byte[] key, int offset)

Crea un objeto DESKeySpec utilizando los primeros 8 bytes en la clave, comenzando en el desplazamiento incluido como offset, como material para la clave DES.

Metodos

Devuelve Tipo

Método y Descripción

byte[]

getKey() Devuelve la clave de cifrado DES.

```
SecretKeyFactory keyfac = SecretKeyFactory.getInstance("DES");
```

```
DESKeySpec keyspec = (DESKeySpec) keyfac.getKeySpec(key, DESKeySpec.class);
```

```
cos.write(keyspec.getKey()); //aquí obtengo la clave que se guarda en fichero
```


Clase *Cipher*

- ▶ Los objetos de la clase ***Cipher*** (*javax.crypto.Cipher*) representan **funciones de cifrado o descifrado**.
- ▶ Se crean especificando el algoritmo que se desea utilizar.
- ▶ Posteriormente pueden ser configurados para realizar tanto operaciones de cifrado como descifrado, indicando en el proceso la clave necesaria.



Clase Cipher

Método	Retorno	Descripción
getInstance(String transformation)	Static Cipher	Método estático para crear objetos de clase Cipher . Recibe como parámetro el nombre del algoritmo de cifrado/descifrado que se desea emplear
Init(int opmode, Key key)	Void	Configura el objeto para que realice operaciones. Los modos de funcionamiento más habituales son Cipher.ENCRYPT_MODE para encriptar información y Cipher.DECRYPT_MODE para desencriptar. Recibe además como parámetro la clave que se desea utilizar
Dofinal(byte[] input)	Byte[]	Realiza la operación para la que ha sido configurado. Recibe como parámetro la secuencia de bytes que se desea cifrar/descifrar y devuelve el resultado de realizar la transformación correspondiente

Ej de cifrado usando Cipher. Algoritmo DES.

```
import java.security.spec.KeySpec;
import javax.crypto.*;

public class CipherEjemplo {
    public static void main(String[] args) {
        try {
            System.out.println("Obteniendo generador de claves de cifrado DES");
            KeyGenerator keygen = KeyGenerator.getInstance("DES");
            System.out.println("Generando la Clave");
            SecretKey key = keygen.generateKey();
            System.out.println("Obteniendo factoria de claves con cifrado DES");
            Cipher desCipher = Cipher.getInstance("DES");
            System.out.println("Configurando Cipher para encriptar");
            desCipher.init(Cipher.ENCRYPT_MODE, key);
            System.out.println("Preparando el mensaje");
            String mensaje = "Mensaje de prueba";
            System.out.println("Mensaje original: " + mensaje);
            System.out.println("Cifrando el mensaje");
            Byte[] bitcodificado = desCipher.doFinal(mensaje.getBytes());
            String mensajeCifrado = new String(bitcodificado);
            System.out.println("Mensaje Cifrado: " + mensajeCifrado);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Constante cifrado Cipher:
"DES/ECB/PKCS5Padding"
(56)

DESCIFRAR

```
Cipher desCipher = Cipher.getInstance("DES");
```

```
desCipher.init(Cipher.DECRYPT_MODE, key);
```

```
Byte[] bitcodificado
=desCipher.doFinal(mensaje.getBytes());
String mensaje_descifrado = new String(bitcodificado);
```

Ej de cifrado y descifrado simétrico

```
import java.security.*;
import javax.crypto.*;
public class Cifrado Simetrico {

    public static void main(String[] args) {
        try {
            System.out.println("Obteniendo generador de claves con cifrado AES");
            KeyGenerator keygen = KeyGenerator.getInstance("AES");
            System.out.println("Generando clave");
            SecretKey key = keygen.generateKey();
            System.out.println("Obteniendo objeto Cipher con cifrado AES");
            Cipher aesCipher = Cipher.getInstance("AES");
            System.out.println("Configurando Cipher para encriptar");
            aesCipher.init(Cipher.ENCRYPT_MODE, key);
            System.out.println(" Preparando el mensaje");
            String mensaje = "Mensaje que se cifrará con AES";
            System.out.println("Mensaje Original: " + mensaje);
            System.out.println("Cifrando el mensaje");
            byte[] bitcodificado = aesCipher.doFinal(mensaje.getBytes());
            String mensajeCifrado = new String(bitcodificado);
            System.out.println("El mensaje cifrado es" + mensajeCifrado);
            System.out.println("Configurando Cipher para descryptar");
            aesCipher.init(Cipher.DECRYPT_MODE, key);
            System.out.println("Descifrando mensaje");
            byte[] bitcodificado2 = aesCipher.doFinal(bitcodificado);
            String mensaje descifrado = new String(bitcodificado2);
            System.out.println(" Mensaje descryptado: " + mensaje descifrado);
        } catch (Exception e) { e.printStackTrace(); }
    }
}
```

Constante cifrado Cipher:
"AES/CBC/PKCS5Padding"
(128)

Programación de cifrado simétrico

- ▶ La interfaz **SecretKey**, que implementa a su vez la interfaz **Key**, representa claves simétricas (opacas).
- ▶ La clase **KeyGenerator** se usa para generar claves simétricas.
- ▶ La clase **SecretKeyFactory** se emplea como clase de factoría de claves basadas en la interfaz **SecretKey**. Para obtener la versión transparente
- ▶ La clase **SecretKeySpec** implementa la interfaz **KeySpec** y sirve como representación transparente de las claves simétricas.



Ejemplo de Creación de Clave

Creación de una clave. Con algoritmo DES (se verá mas adelante). Además se usa `SecretKey`, que implementa la interfaz `Key`, y la clase `SecretKeyFactory`, diseñada para operar con objetos `SecretKey`.

```
import java.security.spec.KeySpec;
import javax.crypto.*; /* es Cipher; KeyGenerator; SecretKey; SecretKeyFactory; spec.DESKeySpec; */
public class CreaClave {
    public static void main(String[] args) {
        try {
            System.out.println("Obteniendo generador de claves de cifrado DES");
            KeyGenerator keygen = KeyGenerator.getInstance("DES");

            System.out.println("Generando la Clave");
            SecretKey key = keygen.generateKey();

            System.out.println("Obteniendo las claves con cifrado DES");
            SecretKeyFactory keyfac=SecretKeyFactory.getInstance("DES");

            System.out.println("Generando keyspec");
            DESKeySpec keyspec = keyfac.getKeySpec(key, DESKeySpec.class);
            System.out.println (keyspec.getKey()); //Imprimo la clave o la puedo almacenar en un archivo
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

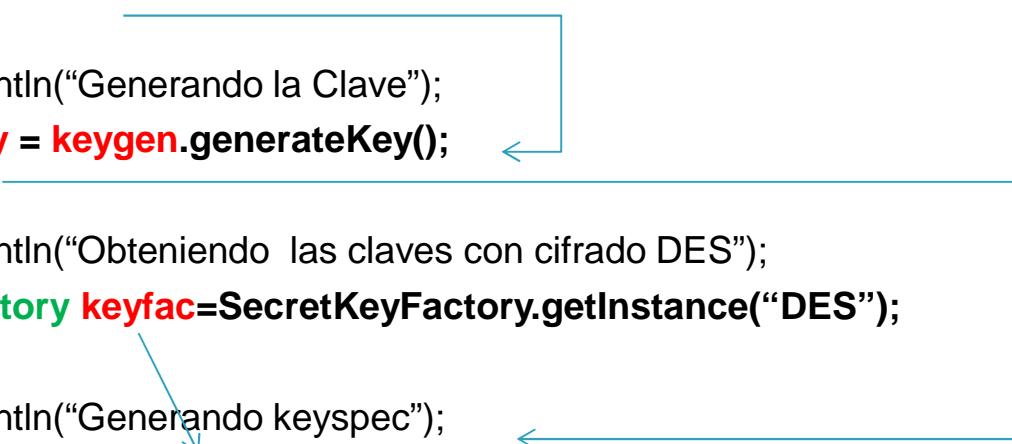


Grafico key y keyspec.



Ejercicio Cifrado

- ▶ Escribe un programa que lea un fichero de texto, lo cifre usando cifrado des y lo escriba en un nuevo fichero.
- ▶ Para ello el programa debe generar una clave usando el objeto generador correspondiente.
- ▶ Una vez el proceso de cifrado haya concluido, el programa debe acceder a la forma transparente de la clave y guardar sus componentes fundamentales en un archivo, de forma que la clave no se pierda y el fichero cifrado pueda ser descifrado en algún momento



Solución

- ▶ La solución de este ejercicio es muy similar al ejemplo visto anteriormente.
- ▶ Usando una clave secreta y un objeto de clase Cipher, ciframos el fichero de prueba, en bloques de 8 bytes.
- ▶ Posteriormente obtenemos la versión transparente de la clave y la almacenamos en otro fichero.
- ▶ Con un programa se cargan los ficheros de salida del primero (datos cifrados y clave) y realiza la operación inversa, descifrando el fichero.



Solución: Código

- ▶ Ver código en 01 – Ejercicio – Fichero Cifrado

