# GENERACIÓN DE LLAVES CRIPTOGRÁFICAS

La Criptografía es casi tan antigua como las civilizaciones humanas:


El término “encriptar” se traduce como “cifrar”. Según la RAE:

CRIPTOGRAFÍA

**encriptar** Conjugar

Del ingl. *to encrypt*; *cf.* gr. ἐγκρύπτειν *enkrýptein* 'ocultar'.

1. tr. **cifrar** (‖ transcribir con una clave).

**cifrar** Conjugar

De *cifra*.

1. tr. Transcribir en guarismos, letras o símbolos, de acuerdo con una clave, un mensaje o texto cuyo contenido se quiere proteger.

El término “criptografía”, según la RAE:

REAL ACADEMIA ESPAÑOLA

Diccionario de la lengua española Edición del Tricentenario Actualización 2022

Consulta posible gracias al compromiso con la cultura de la Fundación "la Caixa"

por palabras | Escriba aquí la palabra | Consultar

**criptografía**

De *cripto-* y *-grafía*.

1. f. Arte de escribir con clave secreta o de un modo enigmático.

A continuación, veamos unos conceptos básicos en el ámbito de la criptografía:

**Mensaje:** Son los datos que actualmente nos conciernen. Normalmente será un texto plano. Denotado por M

**Mensaje cifrado:** Es el mensaje M encriptado. Llamamos al mensaje cifrado C

**Espacio de mensajes:** Son todos los mensajes posibles, en la mayoría de los casos suele ser infinito, pero en otros la longitud puede estar limitada.

**Espacio de mensajes cifrado:** Lo mismo que el punto anterior, pero para mensajes cifrados.

**Espacio de claves:** Son el conjunto de todas las claves posibles. Se representa por la letra K mayúscula. La letra k minúscula representa una clave en concreto, una instancia. Un sinónimo del concepto "clave" es "llave" porque viene del término "key" en inglés. Es por ello que se representa con la letra "K".

Matemáticamente, la encriptación no es más que una aplicación desde el dominio de M al rango de C, y la desencriptación es la función inversa.

Expresamos la encriptación como C = E (M) y la desencriptación como M = D(C)

Y el mensaje se obtiene con la siguiente ecuación: M= D(E(M))

Un ejemplo de lo anterior: Un mensaje M = "hola", una función de encriptación sería escribir la **siguiente letra** del alfabeto, Y una función de desencriptado escribir **la letra anterior** del alfabeto nos quedaría.

M ="hola", C= E(M) ="ipmb", M = D(E(M)) = "hola".

**Criptosistemas de clave privada (algoritmos simétricos).**
Normalmente no se suele usar un método tan sencillo de cifrado, ya que este es lineal y solo depende de un factor (Una entrada siempre produce la misma salida). Un paso más allá es usar una clave/llave para realizar el cifrado, denotándolo así:

Expresamos la encriptación como C = E {k}(M)
La desencriptación como M = D{k}(C)
M= D{k} (E {k} ( M ) )

Pero no es lo mismo cifrar y descifrar con dos claves distintas.
Entonces M != D{k1}(E{k2}(M), donde k1 != k2 ----→ **ALGORITMO ASIMÉTRICO!!!**

**Criptografía con Java**

Desde la aparición de JDK 1.4, Java nos ofrece la posibilidad de trabajar con un framework para criptografía incluido en la JVM. El framework JCE (Java Cryptography Extension) nos ofrece las siguientes características:

- Soporte para cifrado simétrico (DES, RC2, y IDEA)
- Soporte para cifrado simétrico (RSA)
- Cifrado por bloques o por flujo
- Algoritmos MAC (Message Authentication Code)
- Funciones de resumen (funciones hash como MD5 y SHA1 )
- Generación de claves
- Encriptación Basada en Password ( PBE ), transforma un password en una clave robusta mediante procesos aleatorios
- Acuerdo de claves: Es un protocolo, para transmisiones entre dos o más partes, mediante el cual se pueden establecer un acuerdo sobre las claves de cifrado sin intercambiar información secreta

JCE fue un framework opcional hasta la versión 1.4 y apareció por primera vez con la llegada de Java 2.

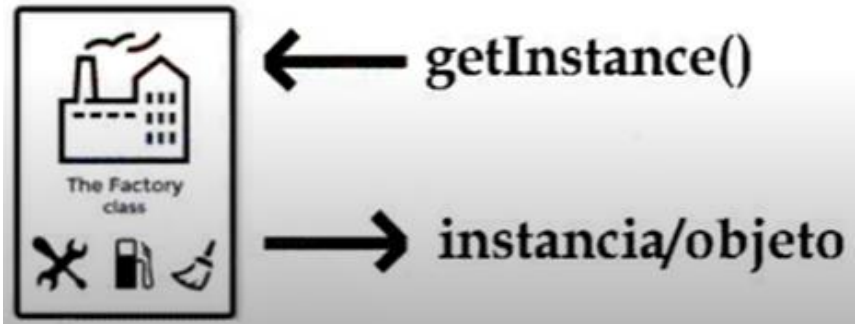
Las clases básicas de JCE son

- Cipher
- Las clases Cipher Stream (CipherInputStream, CipherOutputStream)
- KeyGenerator
- SecretKeyFactory
- KeyAgreement
- Mac

Distribuidas en los siguientes paquetes

- javax.crypto
- javax.crypto.interfaces
- javax.crypto.spec

Las clases de criptografía de Java no generan objetos directamente con new, son clases fábrica que generan objetos llamando a la función/método getInstance():




**Confidencialidad del mensaje con clave privada.**

**Introducción**

Es importante que el atacante no pueda ver el contenido del mensaje que enviemos entre Cliente y Servidor. Veremos cómo cifrar con algoritmos de clave simétrica. JDK 1.4 soporta los siguientes algoritmos:

- DES. DES (Data Encryption Standard) desarrollado por IBM en los 70. Es un cifrado de bloque de 56-bit
- TripleDES. Consiste en aplicar el algoritmo DES tres veces (encriptar desencriptar encriptar) con dos claves dando un resultado de 112 bits.
- AES. Es el algoritmo que reemplazo a DES. Creado por Joan Daemen y Vincent Rijmen. Es un cifrado por bloque de 128-bit con claves de longitud 128, 192 o 256 bits.

**La clase Cipher**

La clase Cipher, se usa para cifrar mediante algoritmos de clave simétrica. Como ya vimos es muy normal instanciar clases a través de una factoría. Este es otro ejemplo, aquí también usaremos getInstance( ):

public static Cipher getInstance(String transformation);

Una transformación tiene la forma:

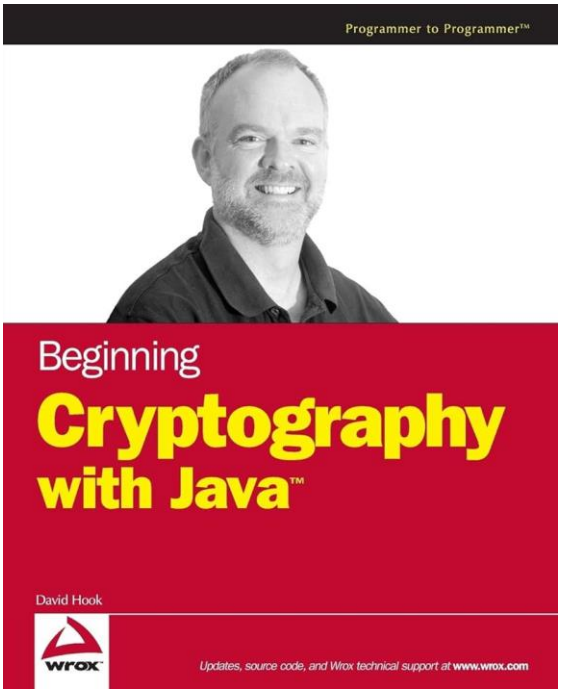
"***algoritmo / modo / relleno*** "

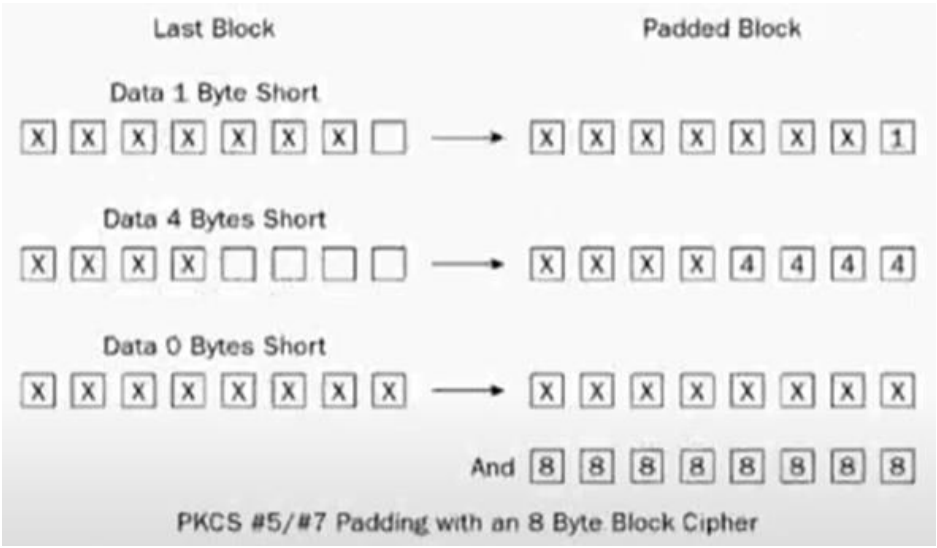
Por ejemplo: Cipher c1 = Cipher.getInstance("*DES/ECB/PKCS5Padding*");

Para utilizar una transformación correcta, revisa la API de Java:

Como hemos visto, un algoritmo puede cifrar por bloques de una longitud determinada, cuando el mensaje es un múltiplo de dicha longitud no existe ningún problema. Pero cuando el mensaje no es un múltiplo, el último bloque es menor que la longitud necesaria para realizar el cifrado, entonces se realiza un relleno de ese bloque. A este relleno se le llama **padding**, Existen varias formas de rellenar los bloques de menor tamaño.

Una fuente que les recomiendo es el siguiente libro:




Este libro explica el padding o relleno PKCS5 que toma el mensaje y lo parte en bloques de 8 bytes, pero si el último bloque no completa los 8 bytes, lo rellena. En síntesis, el algoritmo consiste en rellenar la diferencia para completar 8 bytes y en esos bytes de relleno coloca el valor de la diferencia. En el primer ejemplo es un bloque de 7 bytes, así que rellena con un byte de valor 1. En el segundo ejemplo es un bloque de 4 bytes, así que lo rellena con 4 bytes y en esos 4 bytes el valor es 4. Pero cuando el último bloque está completo con 8 bytes lo rellena con 8 bytes más, con valor 8 en cada uno, por lo que se va a 16 bytes.

**Esquema básico de cifrado con clave privada.**
Veamos un esquema para guiarnos.

**1. Primero crearemos la clave**
**a) KeyGenerator.getInstance("DES")**

**b) .init(56)**
**c) .generateKey()**

2. **Creamos un objeto Cipher y lo configuramos con los parámetros deseados**
**a) Cipher.getInstance("DES/ECB/PKCS5Padding"):**
**b) .init(Cipher.ENCRYPT_MODE, clave):**

**3. Realizamos el cifrado**
**a) .doFinal(textoPlano):**

**4. Configuramos otra vez el objeto Cipher**
**a) .init(Cipher.DECRYPT_MODE, clave)**

**5. Y desciframos**
**a) .doFinal(cipherText)**