19/02/2025

# CRIPTOGRAFÍA APUNTES CLASES

**Conceptos básicos**

- La criptografía es la ciencia que estudia las técnicas para garantizar la seguridad de la información.
- Su objetivo principal es permitir que los datos sean transmitidos y almacenados sin ser accesibles por terceros no autorizados.
- Tradicionalmente, se ha definido como el arte de escribir en clave secreta.
- En la actualidad, es una disciplina matemática que se aplica a la seguridad informática y a las comunicaciones seguras.

**Proceso de encriptación y desencriptación**

- El proceso de encriptar consiste en transformar un mensaje legible (texto claro) en un formato ilegible (texto cifrado).
- A veces, se utiliza una clave para realizar tanto la encriptación como la desencriptación.
- En informática, se emplean términos como "codificar" o "descodificar", y "cifrar" o "descifrar", para referirse a encriptar y desencriptar, respectivamente.
- Ejemplo: Pedro envía un mensaje a Ana a través de internet. Para evitar que otros accedan al contenido, cifra el mensaje con una clave secreta antes de enviarlo. Ana, al recibirlo, usa la misma clave para descifrarlo y leer el mensaje original. Si un tercero intercepta el mensaje sin la clave, solo verá caracteres ilegibles o algo sin sentido.

**Aplicaciones de la criptografía**

- **Identificación y autenticación:** Valida la identidad de usuarios y sistemas.
- **Certificación:** Establece la identidad de un usuario mediante agentes confiables.
- **Seguridad de la comunicación:** Garantiza la confidencialidad de los datos, especialmente en protocolos como HTTPS.
- **Comercio electrónico:** Permite realizar transacciones de manera segura

**Historia de la criptografía.**

- La criptografía tiene antecedentes históricos que se remontan a civilizaciones como los egipcios y los espartanos, quienes la utilizaban para codificar mensajes en guerras.

- Ejemplo de los espartanos: Utilizaban un método ingenioso con un cilindro de un diámetro específico. Enrollaban un papel alrededor del cilindro y escribían el mensaje. Si el papel se desenrollaba, las letras perdían sentido. Para descifrarlo, era necesario volver a enrollarlo en un cilindro del mismo diámetro, lo que reorganizaba el mensaje correctamente.

- En guerras, los ejércitos enviaban mensajes cifrados para que el enemigo no los entendiera al interceptarlos.

**Método de cifrado por desplazamiento (César)**

- Este método consiste en desplazar las letras de un mensaje un número fijo de posiciones en el alfabeto.
- Ejemplo: "Hola" se convierte en "LROD" si se desplaza 3 posiciones (H → K → L, O → R, etc.). La clave es el número de posiciones desplazadas (e.g., 3 o 15).
- Limitación: Este tipo de algoritmos son fáciles de romper, ya que hay un número finito de desplazamientos posibles en el alfabeto. Si se conoce el método, se puede probar con diferentes desplazamientos hasta descifrar el mensaje.

**Características de los servicios de seguridad**

- Para garantizar la seguridad de la información, los servicios criptográficos deben cumplir:
    - **Confidencialidad:** Solo los usuarios autorizados pueden acceder a la información.
    - **Integridad:** Los datos no han sido alterados sin autorización.
    - **Autenticación:** Se verifica la identidad del usuario o sistema.
    - **No repudio:** Evita que un usuario niegue haber realizado una acción específica (e.g., enviar un mensaje).

**Sistema criptográfico: funciones principales**

- Un sistema criptográfico consta de dos funciones principales:
    - **Función de cifrado:** Convierte un mensaje en texto claro a texto cifrado (e.g., desplazamiento en el método César).
    - **Función de descifrado:** Realiza el proceso inverso para recuperar el mensaje original.

**Cifrado simétrico**

- En el cifrado simétrico, se utiliza la misma clave secreta tanto para cifrar como para descifrar
- Ejemplo: Pedro cifra un mensaje para Ana con una clave, y Ana usa esa misma clave para descifrarlo.
- La clave puede ser, por ejemplo, el diámetro del cilindro en el método espartano o el número de posiciones en el cifrado César.
- Existen también cifrados asimétricos (mencionados, pero no detallados aún), donde cada usuario tiene su propia clave.

**Algoritmo AES (Advanced Encryption Standard)**

- Es un algoritmo de cifrado simétrico que protege la información.
- Trabaja con bloques de 128 bits y utiliza claves de 128, 192 o 256 bits.
- Funciona mediante una serie de transformaciones matemáticas en varias iteraciones, incluyendo permutación de filas, mezcla de columnas y combinación con la clave.
- Es rápido, seguro y ampliamente utilizado.

**Herramientas de programación para cifrar en Java**

- Java ofrece clases específicas para implementar criptografía:
    - **Key:** Representa una clave para cifrar.
    - **Spec:** Define especificaciones para las claves y su formato.
    - **Cipher:** Implementa algoritmos de cifrado y descifrado.

- Ejemplo de código

```
// Importamos las clases necesarias para el cifrado simétrico en Java
import javax.crypto.Cipher;// Clase que permite cifrar y descifrar datos
import javax.crypto.KeyGenerator; // Clase para generar claves de cifrado
simétrico
import javax.crypto.SecretKey; // Representa la clave secreta usada en el
cifrado
public class SymmetricEncryption {
 public static void main(String[] args) throws Exception {

 // 1. Generación de una clave secreta AES
// Creamos un generador de claves para AES
 KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
// Establecemos la longitud de la clave en 128 bits (puede ser 192 o 256
también)
 keyGenerator.init(128);
 SecretKey secretKey = keyGenerator.generateKey(); // Generamos la clave AES
 // 2. Creación del objeto Cipher en modo de cifrado
 // Creamos un objeto Cipher configurado para AES
 Cipher cipher = Cipher.getInstance("AES");
 // Inicializamos el cifrador en modo de ENCRIPTACIÓN con la clave generada
 cipher.init(Cipher.ENCRYPT_MODE, secretKey);
 // 3. Definimos el mensaje que queremos cifrar
 String message = "Mensaje seguro"; // Texto original que será cifrado
 // 4. Ciframos el mensaje
 // Convertimos el mensaje en bytes y lo ciframos
 byte[] encryptedMessage = cipher.doFinal(message.getBytes());
 // 5. Inicializamos el Cipher en modo de descifrado
 // Ahora el cifrador se usa para descifrar, utilizando la misma clave
 cipher.init(Cipher.DECRYPT_MODE, secretKey);
 // 6. Desciframos el mensaje
// Desciframos el mensaje cifrado
 byte[] decryptedMessage = cipher.doFinal(encryptedMessage);
 // 7. Mostramos los resultados en consola
 // Mostramos el mensaje original
 System.out.println("Mensaje original: " + message);
// Mostramos el mensaje cifrado (puede no ser legible)
 System.out.println("Mensaje cifrado: " + new String(encryptedMessage));
// Mostramos el mensaje descifrado correctamente
 System.out.println("Mensaje descifrado: " + new String(decryptedMessage));
 }
}
```

20/02/2025

**Algoritmos de Cifrado Simétrico**

- **DES (Data Encryption Standard):** Un algoritmo de cifrado simétrico que utiliza una clave de 56 bits. Aunque fue ampliamente utilizado en el pasado, actualmente se considera vulnerable y no se recomienda su uso.

- **Triple DES:** Una variante de DES que aplica el algoritmo tres veces consecutivas para aumentar la seguridad. Sin embargo, su uso también ha disminuido debido a la disponibilidad de alternativas más seguras.

- **AES (Advanced Encryption Standard):** Un algoritmo de cifrado simétrico que ofrece claves de 128, 192 y 256 bits. Es ampliamente utilizado en la industria debido a su robustez y eficiencia.

**Criptografía Asimétrica (Clave Pública)**

- **Concepto de Clave Pública y Privada:** En la criptografía asimétrica, se utilizan dos claves: una clave pública para cifrar los mensajes y una clave privada para descifrarlos. Esto permite que cualquier persona pueda cifrar un mensaje usando la clave pública del destinatario, pero solo el destinatario con la clave privada correspondiente pueda descifrarlo.

- **Algoritmo RSA:** Uno de los algoritmos más conocidos de criptografía asimétrica. Se basa en la factorización de números primos grandes. La seguridad de RSA radica en la dificultad de factorizar números grandes en sus componentes primos, un problema matemático que actualmente no tiene solución eficiente.

- **Limitaciones Matemáticas:** No existe un algoritmo eficiente para factorizar números primos grandes, lo que hace que RSA sea seguro. Este problema es de interés para la seguridad nacional y está prohibido investigarlo en algunos contextos.

**Firma Digital**

**Concepto de Firma Digital:** Una firma digital permite garantizar la autenticidad e integridad de un mensaje. Se basa en la combinación de un algoritmo de cifrado asimétrico y una función hash.

**Funciones Hash:** Un algoritmo matemático que convierte un mensaje de cualquier longitud en un resumen de tamaño fijo. Las funciones hash son rápidas de calcular, generan valores únicos para entradas distintas y son de un solo sentido (no se puede reconstruir el mensaje original a partir del hash).

**Algoritmos de Firma Digital:**

- **DSA (Digital Signature Algorithm):** Un algoritmo utilizado para firmas digitales.
- **ECDSA (Elliptic Curve Digital Signature Algorithm):** Una variante de DSA que utiliza curvas elípticas para mejorar la eficiencia y la seguridad.

**Implementación Práctica**

- **Generación de Claves:** En el contexto de RSA, se genera un par de claves (pública y privada) utilizando un generador de claves. La longitud de la clave suele ser de 2048 bits para garantizar la seguridad.

- **Cifrado y Descifrado:** Se utiliza la clave pública para cifrar un mensaje y la clave privada para descifrarlo. Este proceso es fundamental en la criptografía asimétrica.

- **Firma y Verificación:** Para firmar un mensaje, se utiliza la clave privada, y para verificar la firma, se utiliza la clave pública. Esto asegura que el mensaje no ha sido alterado y que proviene de la fuente correcta.

**Aplicaciones y Herramientas**

- **Java:** Proporciona herramientas y bibliotecas que simplifican la implementación de algoritmos criptográficos, como la generación de claves, cifrado, descifrado y firma digital.

- **Certificados Digitales:** Utilizados en aplicaciones gubernamentales y empresariales para autenticar la identidad de los usuarios y garantizar la seguridad de las transacciones digitales.

**Criptografía y Curvas Elípticas**

La criptografía de curvas elípticas (ECC) es una alternativa a los métodos tradicionales de criptografía asimétrica. Se basa en las propiedades matemáticas de las curvas elípticas y ofrece un nivel de seguridad similar con claves más cortas, lo que la hace más eficiente.

25/02/2025

**Protocolos de Seguridad en Redes: SSL y TLS**

**SSL (Secure Sockets Layer):** Protocolo de seguridad obsoleto que se utilizaba en las primeras versiones de la web. Permite el cifrado de datos y la comunicación segura entre clientes y servidores. SSL garantiza la autenticación, la integridad y la confidencialidad de los datos.

**TLS (Transport Layer Security):** Protocolo que ha reemplazado a SSL. TLS es el estándar actual para la seguridad en la comunicación entre clientes y servidores. Al igual que SSL, TLS proporciona cifrado de datos, autenticación y garantiza la integridad y confidencialidad de la información.

**Funcionamiento del TLS Handshake**

- Durante el proceso de TLS Handshake, el cliente y el servidor intercambian certificados y establecen una clave de sesión. Esta clave se utiliza para cifrar los datos enviados entre el cliente y el servidor.
- Los certificados y las claves de sesión pueden tener una durabilidad de hasta un año.
- Se utilizan firmas digitales para garantizar la integridad de los datos y autenticar a las partes involucradas en la comunicación.

**Sockets Seguros (SSL/TLS)**

**SSL Server Socket:** Aunque el nombre sugiere el uso de SSL, en realidad las implementaciones modernas utilizan TLS. Las clases como SSLServerSocket y SSLSocket permiten la comunicación cifrada entre aplicaciones distribuidas.

**Puerto 8443:** Es comúnmente utilizado para conexiones seguras mediante SSL/TLS.

**Proceso de Comunicación:**

1. Se crea una fábrica de sockets seguros (`SSLServerSocketFactory`).
2. Se crea un servidor seguro (`SSLServerSocket`) que escucha en un puerto específico (ej. `8443`).
3. El servidor espera la conexión de un cliente.
4. Se crea un buffer para leer los datos enviados por el cliente.
5. El servidor lee el mensaje cifrado y cierra la conexión.

**Control de Acceso**

**Lista de Control de Acceso (ACL):** Define las reglas para que los usuarios puedan acceder a recursos del sistema. Esto incluye permisos a nivel de usuario y roles, donde cada rol tiene permisos específicos.

**Ejemplo de Control de Acceso:**

- Se utiliza la clase `FileAttributeView` para obtener los permisos de un fichero.
- Esto permite verificar los permisos de acceso a un archivo, aunque no se profundiza en la modificación de permisos, ya que esto suele ser responsabilidad del personal de sistemas.