



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TLAXIACO

SEGURIDAD Y VIRTUALIZACIÓN

PRACTICA 1 - CONTRASEÑAS Y CERTIFICADOS

CARRERA:

INGENIERIA EN SISTEMAS COMPUTACIONALES

PRESENTA:

SANDRA YOLOTZIN REYES GARCÍA – 19620079

LUZ KARINA REYES LÓPEZ – 21620184

JERONIMA ROQUE CABALLERO - 21620206

DOCENTE

OSORIO SALINAS EDWARD

Tlaxiaco, Oax., Agosto de 2024.

30/08/2024



“Educación, ciencia y tecnología, progreso día con día” ®

ÍNDICE

PRACTICA 1 - CONTRASEÑAS Y CERTIFICADOS	1
INTRODUCCIÓN.....	4
EJERCICIO 1	5
CODIGO.....	5
EJERCICIO 2	10
CODIGO.....	10
EJERCICIO 3	13
COMANDOS UTILIZADOS.....	17
INVESTIGACIÓN	18
CONTRASEÑA.....	18
CERTIFICADO DIGITAL	18
FIRMA DIGITAL.....	19
CIFRADO SIMÉTRICO.....	19
CIFRADO ASIMÉTRICO.....	20
HASH.....	21
ENCRIPTACIÓN.....	21
¿QUÉ ES LA AES?	22
¿CÓMO FUNCIONA EL CIFRADO AES?	22
¿QUÉ ES RSA?.....	23
¿CÓMO FUNCIONA EL CIFRADO RSA?	24
¿QUÉ ES EL ALGORITMO HASH SHA-256?	24
CÓMO FUNCIONA SHA-256.....	25
SSL	26
CÓMO FUNCIONA SSL	26
VENTAJAS DE SSL	27
¿QUÉ ES TLS?.....	27
CÓMO FUNCIONA TLS.....	28
¿QUÉ ES EL HTTPS?.....	29
¿PARA QUÉ SIRVE EL PROTOCOLO HTTPS?.....	30
SEGURIDAD.....	32
CIFRADO.....	32

SFTP.....	33
¿Cómo funciona SFTP?	33
¿QUÉ ES SSH?.....	34
¿Cómo funciona SSH?	35
CONCLUSIÓN	36

Tabla de ilustración

Ilustración 1 Heru Website 2.0	18
Ilustración 2 Certificado Digital - entenda como funciona essa identidade eletrônica	19
Ilustración 3 Qué es la firma digital y cómo funciona - Carballar.com	19
Ilustración 4 Cifrado simétrico y asimétrico: ¿Cuál es mejor?	20
Ilustración 5 ¿Qué es el cifrado asimétrico? - CriptoMundo	20
Ilustración 6 Hash.....	21
Ilustración 7 ¿QUÉ ES ENCRIPCIÓN? - Conytec.....	21
Ilustración 8 AES-GCM-SIV: Conoce cómo es el nuevo cifrado simétrico AEAD	22
Ilustración 9 Cifrado AES-256 bits, cómo funciona y ¿es realmente seguro?.....	22
Ilustración 10 ¿Qué es el algoritmo AES? KeepCoding Bootcamps.....	23
Ilustración 11 RSA.....	23
Ilustración 12 Clave RSA: ¿qué es y cómo funciona la criptografía RSA? - IONOS España	24
Ilustración 13 SHA-256: The Cryptographic Hash Algorithm Explained.....	25
Ilustración 14 Función hash: concepto y aplicación en Bitcoin Bit2Me Academy	25
Ilustración 15 Qué es el SSL y para qué sirve este certificado	26
Ilustración 16 ¿Cómo funcionan los certificados SSL: guía completa para principiantes	26
Ilustración 17 Ventajas de contar con un certificado de seguridad SSL - TrueData Consultore.....	27
Ilustración 18 ¿QUÉ ES TLS? - Grupo ASICA	27
Ilustración 19 ¿Qué es TLS y cómo funciona? - Espacios Hosting.....	28
Ilustración 20 Qué es TLS y cómo funciona 【Guía completa】	29
Ilustración 21 ¿Qué es el protocolo HTTPS y que ventajas aporta en tu web?	29
Ilustración 22 Que es el Protocolo HTTPS Definición y características	30
Ilustración 23 ¿Qué es el protocolo HTTPS y para qué sirve?	30
Ilustración 24 Http y Https: ¿Qué son y para qué sirven? - 1456 Blog	31
Ilustración 25 seguridad-https - Creamos tu video para empresas	32
Ilustración 26 HTTPS: Cifrado en la web AEPD	33
Ilustración 27 Protocolo SFTP: o que é e para que serve.....	33
Ilustración 28 Aprende qué es SFTP y cómo funciona	34
Ilustración 29 ¿Qué es SSH?	34
Ilustración 30 ¿Cómo funciona el SSH? (2023)	35

INTRODUCCIÓN

En la era digital actual, la seguridad de la información es esencial para proteger datos sensibles y mantener la integridad de las comunicaciones en línea. Las contraseñas y los certificados digitales son herramientas fundamentales en este ámbito, sirviendo como barreras contra accesos no autorizados y garantizando la autenticidad de las transacciones digitales.

Las contraseñas, como primera línea de defensa, deben cumplir con ciertos criterios de seguridad para ser efectivas, incluyendo una combinación adecuada de caracteres y la ausencia de patrones predecibles. Por su parte, los certificados digitales, junto con las firmas digitales, validan la identidad de los participantes en una comunicación, asegurando que los datos transmitidos provienen de una fuente confiable y no han sido manipulados.

Esta práctica tiene como objetivo que los estudiantes comprendan y apliquen conceptos clave de seguridad en la información, como la creación de contraseñas seguras y la implementación de certificados digitales. A través de la programación en Python, la generación de certificados SSH y SSL, y la investigación sobre algoritmos y protocolos de seguridad, los estudiantes desarrollarán habilidades fundamentales para proteger entornos digitales.

EJERCICIO 1

Crea un programa en Python que permita al usuario ingresar una contraseña y que valide si la contraseña es segura o no. Una contraseña segura debe cumplir con los siguientes criterios basados en las recomendaciones de Google:

Tener al menos 8 caracteres.

Tener al menos una letra mayúscula (AZ).

Tener al menos una letra minúscula (az).

Tener al menos un número (0-9).

Tener al menos un carácter especial (! , @ , # , \$, % , ^ , & , * , (,) , , , , , , , , , , - , , , , , , , , , ,) . _ = + [] { } \ | ; : " ' , . < > / ? ~ `

No debe contener espacios en blanco.

No debe tener más de 2 caracteres iguales consecutivos.

Si la contraseña cumple con los criterios, el programa deberá mostrar un mensaje indicando que la contraseña es segura, de lo contrario, deberá mostrar un mensaje indicando que la contraseña no es segura.

CODIGO

```
import java.util.Scanner;
```

/ **

*

* @author sandra yolotzin

*/

```
public class EJERCICIO1_SEGURIDADYVIRTUALIZACION {
```

```
/**
 * @param args the command line arguments
 */

public static void main(String[] args) {
    // Creamos un objeto Scanner para leer la entrada del usuario
    Scanner scanner = new Scanner(System.in);

    // Pedimos al usuario que ingrese una contraseña
    System.out.println("Ingresa tu contraseña: ");
    String password = scanner.nextLine();

    // Verificamos si la contraseña es segura utilizando el método
    isSecurePassword

    if (isSecurePassword(password)) {
        // Si la contraseña es segura, se muestra un mensaje de confirmación
        System.out.println("La contraseña es segura.");
    } else {
        // Si la contraseña no es segura, se mostrará un mensaje de advertencia
        con los criterios que debe cumplir

        System.out.println("La contraseña no es segura. Asegúrate de que cumpla
        los siguientes requisitos:");

        System.out.println("- Al menos 8 caracteres");
        System.out.println("- Al menos una letra mayúscula");
        System.out.println("- Al menos una letra minúscula");
        System.out.println("- Al menos un número");
        System.out.println("- Al menos un carácter especial (@, #, $, etc.)");
        System.out.println("- No debe contener espacios en blanco");
    }
}
```

```
        System.out.println("- No debe tener letras iguales consecutivas");
    }

    scanner.close();
}

// Método que verifica si la contraseña es segura según los criterios dados
public static boolean isSecurePassword(String password) {
    if (password.length() < 8) {
        return false;
    }

    // Variables booleanas para verificar los diferentes criterios de seguridad
    boolean hasUpperCase = false;
    boolean hasLowerCase = false;
    boolean hasDigit = false;
    boolean hasSpecialChar = false;
    boolean hasNoSpaces = !password.contains(" ");

    for (int i = 0; i < password.length(); i++) {
        char c = password.charAt(i);

        if (Character.isUpperCase(c)) {
            hasUpperCase = true;
        } else if (Character.isLowerCase(c)) {
            hasLowerCase = true;
        } else if (Character.isDigit(c)) {
            hasDigit = true;
        }
    }
}
```

```

    } else if ("@#%$%^&+=!".contains(String.valueOf(c))) {
        hasSpecialChar = true;
    }

    // Verificar si hay letras iguales consecutivas
    if (i > 0 && password.charAt(i) == password.charAt(i - 1)) {
        return false;
    }
}

return hasUpperCase && hasLowerCase && hasDigit && hasSpecialChar &&
hasNoSpaces;
}
}

```

Este programa en Java permite al usuario ingresar una contraseña y verifica si es segura basándose en criterios de seguridad específicos. Utiliza el método `isSecurePassword` para asegurar que la contraseña tenga al menos 8 caracteres, incluya una letra mayúscula, una letra minúscula, un número, un carácter especial, no contenga espacios en blanco y no tenga caracteres consecutivos iguales. Si la contraseña cumple con estos requisitos, se indica que es segura; de lo contrario, se muestran los criterios que no cumple.


```

ERJICIO1_SEGURIDADYVIRTUALIZACION - NetBeans IDE 8.1
Archivo Editar Ver Navegar Fuente Reestructurar Ejecutar Depurar Perfil Team Herramientas Ventana Ayuda

P gina de Inicio
Source History
<default config>
import java.util.Scanner;

public class EJERCICIO1_SEGURIDADYVIRTUALIZACION {

    public static void main(String[] args) {
        // Crear un objeto Scanner para leer la entrada del usuario
        Scanner scanner = new Scanner(System.in);

        // Pedir al usuario que ingrese una contrase a
        System.out.println("Ingresa la contrase a: ");
        String password = scanner.nextLine();

        // Verificar si la contrase a se ingres  correctamente utilizando el m todo isSecurePassword
        if (isSecurePassword(password)) {
            // Si la contrase a es segura, se mostrar  un mensaje de confirmaci n
            System.out.println("La contrase a es segura.");
        } else {
            // Si la contrase a no es segura, se mostrar  un mensaje de advertencia con los criterios que debe cumplir
            System.out.println("La contrase a no es segura. Responde de que cumple los siguientes requisitos:");
            System.out.println("- Al menos 8 caracteres");
            System.out.println("- Al menos una letra may scula");
            System.out.println("- Al menos un n mero");
            System.out.println("- Al menos un car cter especial (&, @, $, %, etc.);");
            System.out.println("- No debe haber espacios en blanco");
            System.out.println("- No debe haber letras iguales consecutivas");
        }
    }

    // M todo que verifica si la contrase a es segura seg n los criterios antes
    public static boolean isSecurePassword(String password) {
        if (password.length() < 8)
            return false;

        // Verificar si cumple con los requisitos de seguridad
        boolean hasUpperCase = false;
        boolean hasLowerCase = false;
        boolean hasDigit = false;
        boolean hasSpecialChar = false;
        boolean hasNoSpaces = !password.contains(" ");

        for (int i = 0; i < password.length(); i++) {
            char c = password.charAt(i);

            if (Character.isUpperCase(c)) {
                hasUpperCase = true;
            } else if (Character.isLowerCase(c)) {
                hasLowerCase = true;
            } else if (Character.isDigit(c)) {
                hasDigit = true;
            } else if ("!@#$%^&*~".contains(String.valueOf(c))) {
                hasSpecialChar = true;
            }

            // Verificar si hay letras iguales consecutivas
            if (i > 0 && password.charAt(i) == password.charAt(i - 1)) {
                return false;
            }
        }

        return hasUpperCase && hasLowerCase && hasDigit && hasSpecialChar && hasNoSpaces;
    }
}

```

```

ERJICIO1_SEGURIDADYVIRTUALIZACION - NetBeans IDE 8.1
Archivo Editar Ver Navegar Fuente Reestructurar Ejecutar Depurar Perfil Team Herramientas Ventana Ayuda

P gina de Inicio
Source History
<default config>
import java.util.Scanner;

public class EJERCICIO1_SEGURIDADYVIRTUALIZACION {

    public static void main(String[] args) {
        // Crear un objeto Scanner para leer la entrada del usuario
        Scanner scanner = new Scanner(System.in);

        // Pedir al usuario que ingrese una contrase a
        System.out.println("Ingresa la contrase a: ");
        String password = scanner.nextLine();

        // Verificar si la contrase a se ingres  correctamente utilizando el m todo isSecurePassword
        if (isSecurePassword(password)) {
            // Si la contrase a es segura, se mostrar  un mensaje de confirmaci n
            System.out.println("La contrase a es segura.");
        } else {
            // Si la contrase a no es segura, se mostrar  un mensaje de advertencia con los criterios que debe cumplir
            System.out.println("La contrase a no es segura. Responde de que cumple los siguientes requisitos:");
            System.out.println("- Al menos 8 caracteres");
            System.out.println("- Al menos una letra may scula");
            System.out.println("- Al menos un n mero");
            System.out.println("- Al menos un car cter especial (&, @, $, %, etc.);");
            System.out.println("- No debe haber espacios en blanco");
            System.out.println("- No debe haber letras iguales consecutivas");
        }
    }

    // M todo que verifica si la contrase a es segura seg n los criterios antes
    public static boolean isSecurePassword(String password) {
        if (password.length() < 8)
            return false;

        // Verificar si cumple con los requisitos de seguridad
        boolean hasUpperCase = false;
        boolean hasLowerCase = false;
        boolean hasDigit = false;
        boolean hasSpecialChar = false;
        boolean hasNoSpaces = !password.contains(" ");

        for (int i = 0; i < password.length(); i++) {
            char c = password.charAt(i);

            if (Character.isUpperCase(c)) {
                hasUpperCase = true;
            } else if (Character.isLowerCase(c)) {
                hasLowerCase = true;
            } else if (Character.isDigit(c)) {
                hasDigit = true;
            } else if ("!@#$%^&*~".contains(String.valueOf(c))) {
                hasSpecialChar = true;
            }

            // Verificar si hay letras iguales consecutivas
            if (i > 0 && password.charAt(i) == password.charAt(i - 1)) {
                return false;
            }
        }

        return hasUpperCase && hasLowerCase && hasDigit && hasSpecialChar && hasNoSpaces;
    }
}

```

```

ERJICIO1_SEGURIDADYVIRTUALIZACION - NetBeans IDE 8.1
Archivo Editar Ver Navegar Fuente Reestructurar Ejecutar Depurar Perfil Team Herramientas Ventana Ayuda

P gina de Inicio
Source History
<default config>
import java.util.Scanner;

public class EJERCICIO1_SEGURIDADYVIRTUALIZACION {

    public static void main(String[] args) {
        // Crear un objeto Scanner para leer la entrada del usuario
        Scanner scanner = new Scanner(System.in);

        // Pedir al usuario que ingrese una contrase a
        System.out.println("Ingresa la contrase a: ");
        String password = scanner.nextLine();

        // Verificar si la contrase a se ingres  correctamente utilizando el m todo isSecurePassword
        if (isSecurePassword(password)) {
            // Si la contrase a es segura, se mostrar  un mensaje de confirmaci n
            System.out.println("La contrase a es segura.");
        } else {
            // Si la contrase a no es segura, se mostrar  un mensaje de advertencia con los criterios que debe cumplir
            System.out.println("La contrase a no es segura. Responde de que cumple los siguientes requisitos:");
            System.out.println("- Al menos 8 caracteres");
            System.out.println("- Al menos una letra may scula");
            System.out.println("- Al menos un n mero");
            System.out.println("- Al menos un car cter especial (&, @, $, %, etc.);");
            System.out.println("- No debe haber espacios en blanco");
            System.out.println("- No debe haber letras iguales consecutivas");
        }
    }

    // M todo que verifica si la contrase a es segura seg n los criterios antes
    public static boolean isSecurePassword(String password) {
        if (password.length() < 8)
            return false;

        // Verificar si cumple con los requisitos de seguridad
        boolean hasUpperCase = false;
        boolean hasLowerCase = false;
        boolean hasDigit = false;
        boolean hasSpecialChar = false;
        boolean hasNoSpaces = !password.contains(" ");

        for (int i = 0; i < password.length(); i++) {
            char c = password.charAt(i);

            if (Character.isUpperCase(c)) {
                hasUpperCase = true;
            } else if (Character.isLowerCase(c)) {
                hasLowerCase = true;
            } else if (Character.isDigit(c)) {
                hasDigit = true;
            } else if ("!@#$%^&*~".contains(String.valueOf(c))) {
                hasSpecialChar = true;
            }

            // Verificar si hay letras iguales consecutivas
            if (i > 0 && password.charAt(i) == password.charAt(i - 1)) {
                return false;
            }
        }

        return hasUpperCase && hasLowerCase && hasDigit && hasSpecialChar && hasNoSpaces;
    }
}

```

```

P gina de Inicio
Source History
<default config>
import java.util.Scanner;

public class EJERCICIO1_SEGURIDADYVIRTUALIZACION {

    public static void main(String[] args) {
        // Crear un objeto Scanner para leer la entrada del usuario
        Scanner scanner = new Scanner(System.in);

        // Pedir al usuario que ingrese una contrase a
        System.out.println("Ingresa la contrase a: ");
        String password = scanner.nextLine();

        // Verificar si la contrase a se ingres  correctamente utilizando el m todo isSecurePassword
        if (isSecurePassword(password)) {
            // Si la contrase a es segura, se mostrar  un mensaje de confirmaci n
            System.out.println("La contrase a es segura.");
        } else {
            // Si la contrase a no es segura, se mostrar  un mensaje de advertencia con los criterios que debe cumplir
            System.out.println("La contrase a no es segura. Responde de que cumple los siguientes requisitos:");
            System.out.println("- Al menos 8 caracteres");
            System.out.println("- Al menos una letra may scula");
            System.out.println("- Al menos un n mero");
            System.out.println("- Al menos un car cter especial (&, @, $, %, etc.);");
            System.out.println("- No debe haber espacios en blanco");
            System.out.println("- No debe haber letras iguales consecutivas");
        }
    }

    // M todo que verifica si la contrase a es segura seg n los criterios antes
    public static boolean isSecurePassword(String password) {
        if (password.length() < 8)
            return false;

        // Verificar si cumple con los requisitos de seguridad
        boolean hasUpperCase = false;
        boolean hasLowerCase = false;
        boolean hasDigit = false;
        boolean hasSpecialChar = false;
        boolean hasNoSpaces = !password.contains(" ");

        for (int i = 0; i < password.length(); i++) {
            char c = password.charAt(i);

            if (Character.isUpperCase(c)) {
                hasUpperCase = true;
            } else if (Character.isLowerCase(c)) {
                hasLowerCase = true;
            } else if (Character.isDigit(c)) {
                hasDigit = true;
            } else if ("!@#$%^&*~".contains(String.valueOf(c))) {
                hasSpecialChar = true;
            }

            // Verificar si hay letras iguales consecutivas
            if (i > 0 && password.charAt(i) == password.charAt(i - 1)) {
                return false;
            }
        }

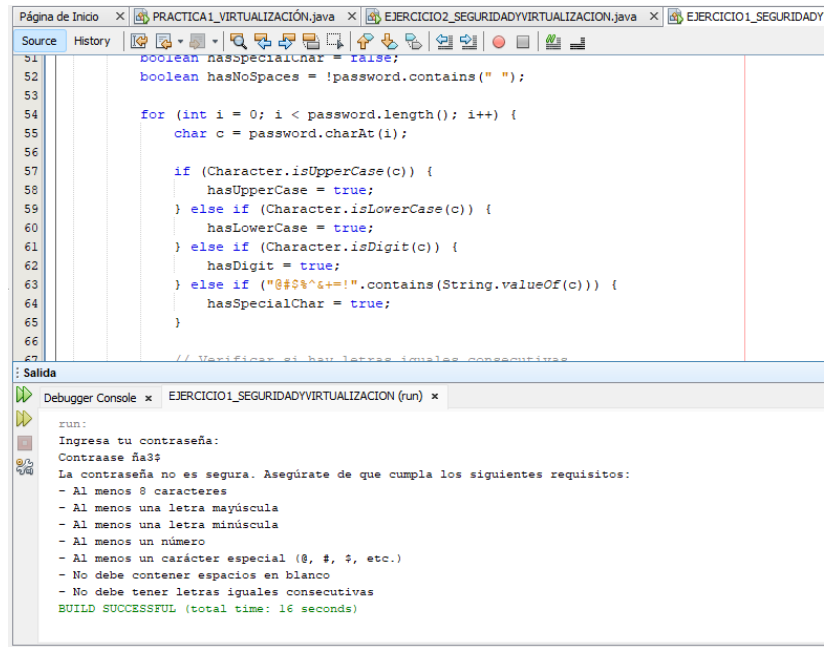
        return hasUpperCase && hasLowerCase && hasDigit && hasSpecialChar && hasNoSpaces;
    }
}

```

```

run:
Ingresa tu contrase a:
Contrase a79#
La contrase a es segura.
BUILD SUCCESSFUL (total time: 15 seconds)

```



EJERCICIO 2

Crea un programa que me recomiende una contraseña segura. La contraseña debe cumplir con los criterios de la instrucción anterior.

CODIGO

```
import java.util.Random;
```

```
/**
```

```
*
```

```
* @author sandra yolotzin
```

```
*/
```

```
public class EJERCICIO2_SEGURIDADYVIRTUALIZACION {
```

```
/**
```

```
* @param args the command line arguments
```

```
*/
```

```

public static void main(String[] args) {
    String securePassword = generateSecurePassword(12); // Se puede cambiar
    el tamaño de la contraseña si es necesario

    System.out.println("Contraseña segura recomendada: " + securePassword);
}

public static String generateSecurePassword(int length) {
    String upperCaseLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    String lowerCaseLetters = "abcdefghijklmnopqrstuvwxyz";
    String digits = "0123456789";
    String specialCharacters = "@#$%^&+=!";

    String allCharacters = upperCaseLetters + lowerCaseLetters + digits +
    specialCharacters;

    Random random = new Random();
    StringBuilder password = new StringBuilder();

    // Asegurar que la contraseña contenga al menos un carácter de cada tipo

    password.append(upperCaseLetters.charAt(random.nextInt(upperCaseLetters.length())));

    password.append(lowerCaseLetters.charAt(random.nextInt(lowerCaseLetters.length())));

    password.append(digits.charAt(random.nextInt(digits.length())));

    password.append(specialCharacters.charAt(random.nextInt(specialCharacters.length())));

    // Rellenar el resto de la contraseña con caracteres aleatorios

```

```

        for (int i = 4; i < length; i++) {
            char nextChar;
            do {
                nextChar = allCharacters.charAt(random.nextInt(allCharacters.length()));
            } while (i > 0 && nextChar == password.charAt(i - 1)); // Evitar caracteres
            iguales consecutivos
            password.append(nextChar);
        }

        // Mezclar los caracteres para mayor aleatoriedad
        return shuffleString(password.toString());
    }

    public static String shuffleString(String input) {
        char[] characters = input.toCharArray();
        Random random = new Random();
        for (int i = 0; i < characters.length; i++) {
            int randomIndex = random.nextInt(characters.length);
            char temp = characters[i];
            characters[i] = characters[randomIndex];
            characters[randomIndex] = temp;
        }
        return new String(characters);
    }
}

```

Este programa en Java genera una contraseña segura recomendada con un tamaño especificado, asegurando que incluya al menos una letra mayúscula, una letra minúscula, un número y un carácter especial. Utiliza el método **`generateSecurePassword`** para crear una contraseña que cumple con estos requisitos y evita caracteres iguales consecutivos. Luego, la contraseña generada es mezclada aleatoriamente mediante el método **`shuffleString`** para aumentar la

aleatoriedad. El resultado se muestra al usuario como una contraseña segura recomendada.

```
ERECICIO2_SEGURIDADYVIRTUALIZACION - NetBeans IDE 8.1
Archivo Editor View Herramientas Fuente Reestructura Ejecutar Depurar Perfil Test Herramientas Ventana Ayuda

// Ejercicio 2: Generar una contraseña segura recomendada
// Autor: [Nombre]
// Fecha: [Fecha]

import java.util.Random;

/**
 * Clase que genera una contraseña segura recomendada.
 */
public class Ejercicio2_SEGURIDADYVIRTUALIZACION {

    // Constantes para la longitud de la contraseña y los caracteres permitidos
    private static final int LONGITUD = 12;
    private static final String upperCaseLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    private static final String lowerCaseLetters = "abcdefghijklmnopqrstuvwxyz";
    private static final String digits = "0123456789";
    private static final String specialCharacters = "!@#$%^&*~.-_+={}[]\';\"<,>?/\\|:~";

    // Método principal para generar la contraseña
    public static void main(String[] args) {
        String securePassword = generateSecurePassword(LONGITUD); // Se genera la contraseña al tamaño de la contraseña si se proporciona
        System.out.println("Contraseña segura recomendada: " + securePassword);
    }

    // Método para generar la contraseña
    public static String generateSecurePassword(int length) {
        String password = "";
        Random random = new Random();
        StringBuilder passwordBuilder = new StringBuilder();

        // Asegurar que la contraseña contenga al menos un carácter de cada tipo
        passwordBuilder.append(upperCaseLetters.charAt(random.nextInt(upperCaseLetters.length())));
        passwordBuilder.append(lowerCaseLetters.charAt(random.nextInt(lowerCaseLetters.length())));
        passwordBuilder.append(digits.charAt(random.nextInt(digits.length())));
        passwordBuilder.append(specialCharacters.charAt(random.nextInt(specialCharacters.length())));

        // Rellenar el resto de la contraseña con caracteres aleatorios
        for (int i = 4; i < length; i++) {
            char nextChar;
            do {
                nextChar = allCharacters.charAt(random.nextInt(allCharacters.length()));
            } while (i > 0 && nextChar == password.charAt(i - 1)); // Evitar caracteres iguales
            passwordBuilder.append(nextChar);
        }

        // Mezclar los caracteres para mayor aleatoriedad
        return shuffleString(passwordBuilder.toString());
    }

    // Método para mezclar los caracteres
    private static String shuffleString(String input) {
        char[] characters = input.toCharArray();
        Random random = new Random();
        for (int i = 0; i < characters.length; i++) {
            int randomIndex = random.nextInt(characters.length);
            char temp = characters[i];
            characters[i] = characters[randomIndex];
            characters[randomIndex] = temp;
        }
        return new String(characters);
    }
}
```

```
ERECICIO2_SEGURIDADYVIRTUALIZACION - NetBeans IDE 8.1
Archivo Editor View Herramientas Fuente Reestructura Ejecutar Depurar Perfil Test Herramientas Ventana Ayuda

// Ejercicio 2: Generar una contraseña segura recomendada
// Autor: [Nombre]
// Fecha: [Fecha]

import java.util.Random;

/**
 * Clase que genera una contraseña segura recomendada.
 */
public class Ejercicio2_SEGURIDADYVIRTUALIZACION {

    // Constantes para la longitud de la contraseña y los caracteres permitidos
    private static final int LONGITUD = 12;
    private static final String upperCaseLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    private static final String lowerCaseLetters = "abcdefghijklmnopqrstuvwxyz";
    private static final String digits = "0123456789";
    private static final String specialCharacters = "!@#$%^&*~.-_+={}[]\';\"<,>?/\\|:~";

    // Método principal para generar la contraseña
    public static void main(String[] args) {
        String securePassword = generateSecurePassword(LONGITUD); // Se genera la contraseña al tamaño de la contraseña si se proporciona
        System.out.println("Contraseña segura recomendada: " + securePassword);
    }

    // Método para generar la contraseña
    public static String generateSecurePassword(int length) {
        String password = "";
        Random random = new Random();
        StringBuilder passwordBuilder = new StringBuilder();

        // Asegurar que la contraseña contenga al menos un carácter de cada tipo
        passwordBuilder.append(upperCaseLetters.charAt(random.nextInt(upperCaseLetters.length())));
        passwordBuilder.append(lowerCaseLetters.charAt(random.nextInt(lowerCaseLetters.length())));
        passwordBuilder.append(digits.charAt(random.nextInt(digits.length())));
        passwordBuilder.append(specialCharacters.charAt(random.nextInt(specialCharacters.length())));

        // Rellenar el resto de la contraseña con caracteres aleatorios
        for (int i = 4; i < length; i++) {
            char nextChar;
            do {
                nextChar = allCharacters.charAt(random.nextInt(allCharacters.length()));
            } while (i > 0 && nextChar == password.charAt(i - 1)); // Evitar caracteres iguales
            passwordBuilder.append(nextChar);
        }

        // Mezclar los caracteres para mayor aleatoriedad
        return shuffleString(passwordBuilder.toString());
    }

    // Método para mezclar los caracteres
    private static String shuffleString(String input) {
        char[] characters = input.toCharArray();
        Random random = new Random();
        for (int i = 0; i < characters.length; i++) {
            int randomIndex = random.nextInt(characters.length);
            char temp = characters[i];
            characters[i] = characters[randomIndex];
            characters[randomIndex] = temp;
        }
        return new String(characters);
    }
}
```

```
Source History
27
28 // Asegurar que la contraseña contenga al menos un carácter de cada tipo
29 password.append(upperCaseLetters.charAt(random.nextInt(upperCaseLetters.length())));
30 password.append(lowerCaseLetters.charAt(random.nextInt(lowerCaseLetters.length())));
31 password.append(digits.charAt(random.nextInt(digits.length())));
32 password.append(specialCharacters.charAt(random.nextInt(specialCharacters.length())));
33
34 // Rellenar el resto de la contraseña con caracteres aleatorios
35 for (int i = 4; i < length; i++) {
36     char nextChar;
37     do {
38         nextChar = allCharacters.charAt(random.nextInt(allCharacters.length()));
39     } while (i > 0 && nextChar == password.charAt(i - 1)); // Evitar caracteres iguales
40     password.append(nextChar);
41 }
42
43 // Mezclar los caracteres para mayor aleatoriedad
44 return shuffleString(passwordBuilder.toString());
45 }

Salida
Debugger Console x EJERCICIO2_SEGURIDADYVIRTUALIZACION (run) x
run:
Contraseña segura recomendada: 5G~vex857uSz
BUILD SUCCESSFUL (total time: 0 seconds)
```

EJERCICIO 3

3. Crea un certificado SSH, clave pública y clave privada, añade el certificado SSH a tu cuenta de GitHub y realiza un git clone de un repositorio nuevo utilizando la ruta SSH del repositorio.

- Primero abrimos nuestra terminal en Git Bash

- Luego ejecutamos el siguiente comando `ssh-keygen -t ed25519 -C "sandra.yolotzin.reyes.garcia@gmail.com"` para generar una clave SSH
- Después para añadir la clave SSH a nuestro agente SSH, iniciamos el agente SSH con el comando `eval "$(ssh-agent -s)"`.
- Posteriormente añadimos el comando para obtener nuestra clave privada SSH, que es en este caso

ssh-ed25519

AAAAC3NzaC1lZDI1NTE5AAAAIGh3nFcg7IPcs/otpurioKwWBnWilJHjlgMXSOHQwBX6 sandra.yolotzin.reyes.garcia@gmail.com

como se muestra en la captura

```

MINGW64~/c/Users/sandr
sandr@Sandy MINGW64 ~$ ssh-add ~/.ssh/id_ed25519
Identity added: /c/Users/sandr/.ssh/id_ed25519 (sandra.yolotzin.reyes.garcia@gmail.com)

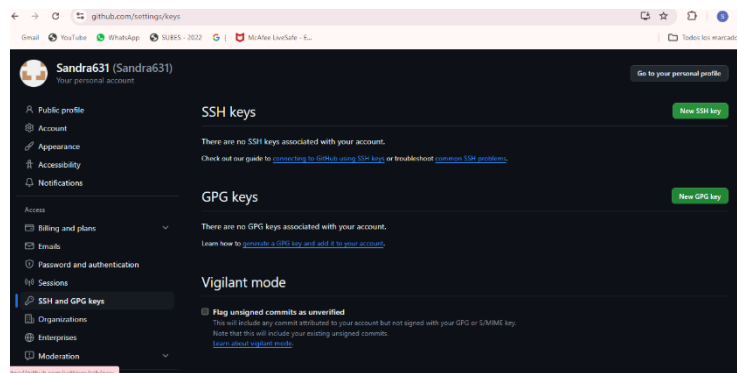
sandr@Sandy MINGW64 ~$ rm ~/.ssh/id_ed25519 ~/.ssh/id_ed25519.pub

sandr@Sandy MINGW64 ~$ ssh-keygen -t ed25519 -C "sandra.yolotzin.reyes.garcia@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/sandr/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/sandr/.ssh/id_ed25519
Your public key has been saved in /c/Users/sandr/.ssh/id_ed25519.pub
The key's fingerprint is:
SHA256:Y1D9P7Y2w+4ZDUC1gN13MQY5w1bxyTed4j/7wSQNkg sandra.yolotzin.reyes.garcia@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
|  .  .  .  .  .  |
| o  .  .  .  .  .  |
| .  .  .  .  .  .  |
| .  .  .  .  .  .  |
|  .  .  .  .  .  .  |
|  .  .  .  .  .  .  |
|  .  .  .  .  .  .  |
|  .  .  .  .  .  .  |
|  .  .  .  .  .  .  |
+-----[SHA256]-----+

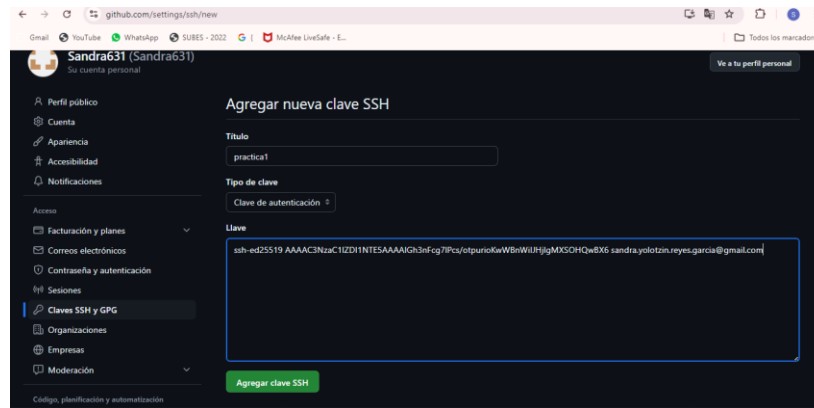
sandr@Sandy MINGW64 ~$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGh3nFcg7IPcs/otpurioKwWBnWilJHjlgMXSOHQwBX6 sandra.yolotzin.reyes.garcia@gmail.com
sandr@Sandy MINGW64 ~$

```

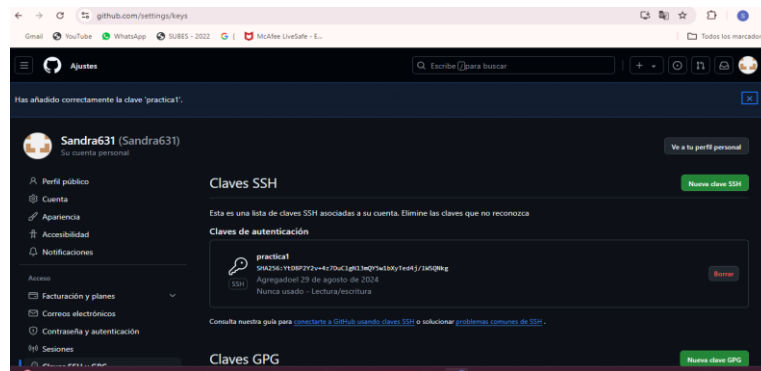
Luego nos vamos a nuestro perfil de GitHub en Settings(configuración), luego a SSH and GPG keys,despues dimos clic en New SSH key.



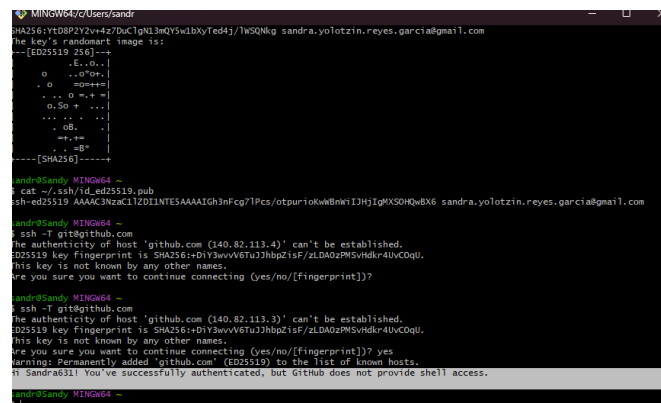
Posteriormente le agregamos un titulo y agregamos nuestra llave que se nos habia generado anteriormente y damos clic en agregar clave SSH.



Después nos muestra un mensaje que nos dice que la clave se ha añadido correctamente

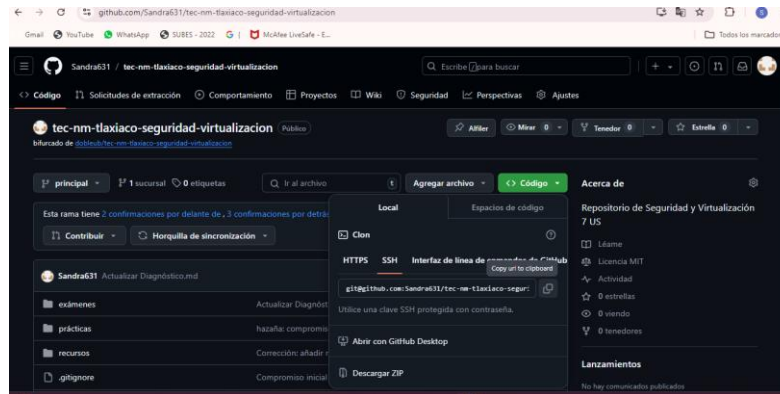


Verificar conectividad para empezar a clonar nuestro repositorio



Luego empezamos a clonar el repositorio que deseamos, Primero entramos al repositorio en GitHub que deseábamos clonar.

Luego dimos clic en el botón verde que dice Code (código) y seleccionamos la opción SSH. Después copiamos la URL SSH del repositorio



Luego agregamos el comando GitHub clone junto con el URL SSH [git@github.com:Sandra631/tec-nm-tlaxiaco-seguridad-virtualizacion.git](https://github.com/Sandra631/tec-nm-tlaxiaco-seguridad-virtualizacion.git) para clonar el repositorio y después se puede observar cómo se empieza a clonar el repositorio

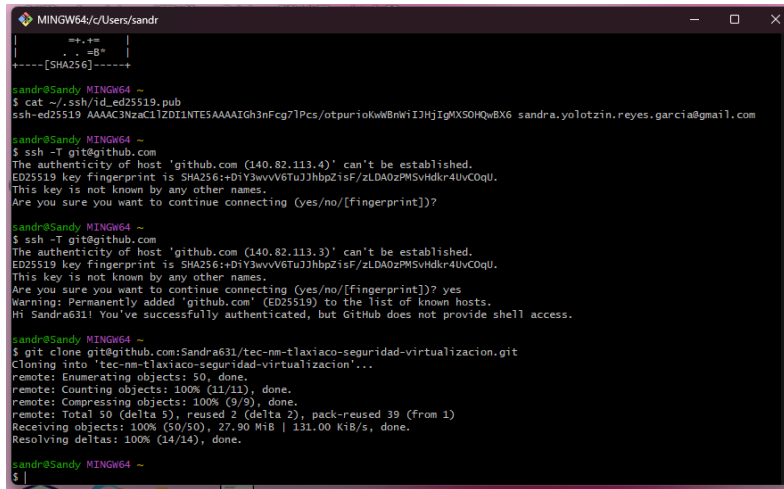
```
MINGW64/c/Users/sandr
+ 0  =0m++=
+ .. 0 =.+ =
+ 0 So + ...
+ ... ..
+ .. 0B.
+ ..+..
+ .. =B*
+ ----[SHA256]-----

sandr@sandy MINGW64 ~
$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZD11NTESAAAAIGH3nFc7TPcs/otpuriokwBnwiIjHjIgMXSOHQwBX6 sandra.yolotzin.reyes.garcia@gmail.com

sandr@sandy MINGW64 ~
$ ssh -T git@github.com
The authenticity of host 'github.com (140.82.113.4)' can't be established.
ED25519 key fingerprint is SHA256:+D1Y3wrvV6TuJhbpZisF/zLDA0zPMSvHdkr4UvCoQg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
sandr@sandy MINGW64 ~
$ ssh -T git@github.com
The authenticity of host 'github.com (140.82.113.3)' can't be established.
ED25519 key fingerprint is SHA256:+D1Y3wrvV6TuJhbpZisF/zLDA0zPMSvHdkr4UvCoQg.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi Sandra631! You've successfully authenticated, but GitHub does not provide shell access.

sandr@sandy MINGW64 ~
$ git clone git@github.com:Sandra631/tec-nm-tlaxiaco-seguridad-virtualizacion.git
Cloning into 'tec-nm-tlaxiaco-seguridad-virtualizacion'...
remote: Enumerating objects: 50, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
Receiving objects: 48% (24/50), 2.68 MiB | 268.00 KiB/s
```


Finalmente se observa que termino de clonarse exitosamente el repositorio



```
MINGW64/c/Users/sandr
+-----[SHA256]-----+
sandr@sandy MINGW64 ~
$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZD1lNTESAAAAIGh3nFc7lPcs/otpuriokWbNwIjHjIgMXSOHQwBX6 sandra.yolotzin.reyes.garcia@gmail.com
sandr@sandy MINGW64 ~
$ ssh -T git@github.com
The authenticity of host 'github.com (140.82.113.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6Tu3Jhbp2isF/zLDA0zPM5vHdKr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
sandr@sandy MINGW64 ~
$ ssh -T git@github.com
The authenticity of host 'github.com (140.82.113.3)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvV6Tu3Jhbp2isF/zLDA0zPM5vHdKr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Hi Sandra631! You've successfully authenticated, but GitHub does not provide shell access.
sandr@sandy MINGW64 ~
$ git clone git@github.com:Sandra631/tec-nm-tlaxiaco-seguridad-virtualizacion.git
Cloning into 'tec-nm-tlaxiaco-seguridad-virtualizacion'...
remote: Enumerating objects: 50, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 50 (delta 5); reused 2 (delta 2), pack-reused 39 (from 1)
Receiving objects: 100% (50/50), 27.90 MiB | 131.00 KiB/s, done.
Resolving deltas: 100% (14/14), done.
sandr@sandy MINGW64 ~
$
```

COMANDOS UTILIZADOS

➤ **ssh-keygen -t ed25519 -C " sandra.yolotzin.reyes.garcia@gmail.com "**

-t ed25519: Especifica el tipo de clave. ed25519 es un tipo de clave moderno y seguro.

-C "tu_email@example.com": Añade un comentario con tu email para identificar la clave.

- eval "\$(ssh-agent -s)"
- ssh-add ~/.ssh/id_ed25519
- cat ~/.ssh/id_ed25519.pub
- ssh -T git@github.com
- git clone git@github.com:Sandra631/tec-nm-tlaxiaco-seguridad-virtualizacion.git

INVESTIGACIÓN

CONTRASEÑA

Una contraseña es una secuencia de caracteres que un usuario utiliza para demostrar su identidad al acceder a un sistema, servicio o dispositivo. Las contraseñas son un método de autenticación y control de acceso que protege la información de accesos no autorizados. En la seguridad informática, es importante que las contraseñas sean complejas y difíciles de adivinar, lo que generalmente se logra combinando letras mayúsculas y minúsculas, números y símbolos. Además, las contraseñas deben ser únicas para cada cuenta y cambiarse regularmente para minimizar el riesgo de ser comprometidas.



Ilustración 1 Heru Website 2.0

Las contraseñas también pueden ser almacenadas en formato de hash para mayor seguridad, lo que significa que el sistema no guarda la contraseña en sí, sino una representación cifrada de ella. Si un atacante obtiene acceso a la base de datos de contraseñas, solo verá los hashes, no las contraseñas originales.

CERTIFICADO DIGITAL

Un certificado digital es un documento electrónico que sirve para autenticar la identidad de un usuario, dispositivo o servidor en una red digital. Emitido por una Autoridad Certificadora (CA), el certificado contiene información como el nombre del titular, su clave pública, la firma digital de la CA, y la fecha de expiración del certificado. Este documento permite establecer conexiones seguras a través de protocolos como HTTPS, asegurando que la comunicación entre el cliente y el servidor no sea interceptada por terceros.



Ilustración 2 Certificado Digital - entenda como funciona essa identidade eletrônica ...

Los certificados digitales son fundamentales en la infraestructura de clave pública (PKI) y se utilizan para firmar electrónicamente documentos, cifrar correos electrónicos, y asegurar transacciones en línea. Un certificado válido garantiza al usuario que está interactuando con la entidad legítima.

FIRMA DIGITAL

Una firma digital es un mecanismo de criptografía que asegura que un documento electrónico ha sido creado por una persona específica y que no ha sido alterado desde que se firmó. La firma digital se genera al cifrar un resumen (hash) del documento con la clave privada del firmante. Este resumen cifrado, junto con la clave pública del firmante, permite a los destinatarios verificar la autenticidad y la integridad del documento.



Ilustración 3 Qué es la firma digital y cómo funciona - Carballar.com

Las firmas digitales tienen la misma validez legal que las firmas manuscritas en muchos países y se utilizan para firmar contratos, declaraciones y otros documentos importantes. Además, las firmas digitales proporcionan no repudio, lo que significa que el firmante no puede negar haber firmado el documento.

CIFRADO SIMÉTRICO

El cifrado simétrico es un método de criptografía en el cual la misma clave se utiliza tanto para cifrar como para descifrar datos. Este tipo de cifrado es muy eficiente y rápido, lo que lo hace ideal para proteger grandes volúmenes de datos, como archivos o bases de datos. Sin embargo, uno de los desafíos del cifrado simétrico

es la distribución segura de la clave, ya que cualquier persona que posea la clave puede descifrar la información.

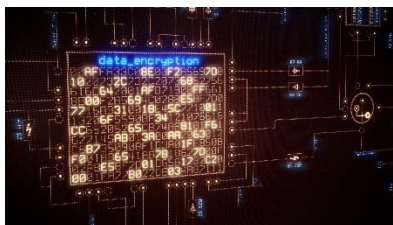


Ilustración 4 Cifrado simétrico y asimétrico: ¿Cuál es mejor?

Ejemplos de algoritmos de cifrado simétrico incluyen el AES (Advanced Encryption Standard), que es ampliamente utilizado en sistemas modernos, y DES (Data Encryption Standard), que fue común en el pasado, pero ha sido reemplazado por AES debido a problemas de seguridad.

CIFRADO ASIMÉTRICO

El cifrado asimétrico es una técnica de criptografía que utiliza un par de claves: una clave pública, que se comparte abiertamente, y una clave privada, que se mantiene en secreto. Un mensaje cifrado con la clave pública solo puede ser descifrado con la clave privada correspondiente, lo que proporciona un método seguro para intercambiar información sin necesidad de compartir una clave secreta de antemano.



Ilustración 5 ¿Qué es el cifrado asimétrico? - CriptoMundo

Este tipo de cifrado es fundamental para el funcionamiento de protocolos de seguridad en internet, como SSL/TLS, que protegen las comunicaciones en línea. Además, el cifrado asimétrico se utiliza para la firma digital, en la que la clave privada del firmante cifra un hash del documento, permitiendo a cualquier persona con la clave pública verificar la firma.

Ejemplos de algoritmos de cifrado asimétrico incluyen RSA y ECC (Elliptic Curve Cryptography).

HASH

Un hash es una función matemática que toma una entrada de cualquier longitud y la convierte en una cadena de longitud fija, que sirve como una "huella digital" única del dato original. Los hashes son deterministas, lo que significa que la misma entrada siempre producirá el mismo hash, pero incluso un pequeño cambio en la entrada producirá un hash completamente diferente.



Ilustración 6 Hash

Los hashes se utilizan en diversas aplicaciones de seguridad, como la verificación de integridad de archivos, almacenamiento seguro de contraseñas, y en las criptomonedas como Bitcoin, donde se utilizan para garantizar la inmutabilidad de las transacciones. Ejemplos de funciones de hash comunes incluyen SHA-256 y MD5. Es importante destacar que los hashes no son reversibles, lo que significa que no es posible obtener la entrada original a partir del hash.

ENCRIPCIÓN

Encriptación es el proceso de convertir datos legibles (texto plano) en un formato codificado (texto cifrado) que solo puede ser leído o descifrado por personas autorizadas que poseen la clave de descifrado. La encriptación es fundamental para proteger la confidencialidad de la información durante la transmisión o el almacenamiento, asegurando que solo las partes autorizadas puedan acceder a los datos.



Ilustración 7 ¿QUÉ ES ENCRIPCIÓN? - Conytec

Existen dos tipos principales de encriptación:

Cifrado simétrico: Utiliza la misma clave para cifrar y descifrar los datos.

Cifrado asimétrico: Utiliza un par de claves (pública y privada) para cifrar y descifrar los datos.

La encriptación se utiliza en diversas aplicaciones, desde la protección de comunicaciones en línea (por ejemplo, HTTPS) hasta el cifrado de dispositivos y archivos. Es una herramienta esencial en la seguridad de la información, protegiendo datos sensibles de accesos no autorizados.

¿QUÉ ES LA AES?

AES, o Advanced Encryption Standard, es un algoritmo de cifrado ampliamente utilizado que garantiza la seguridad de las comunicaciones y la protección de los datos. Funciona con bloques de datos, normalmente de 128 bits, y utiliza una sola clave (la misma) de 128, 192 o 256 bits para cifrar y descifrar la información.



Ilustración 8 AES-GCM-SIV: Conoce cómo es el nuevo cifrado simétrico AEAD

Es un algoritmo simétrico que utiliza un cifrado por bloques y la misma clave de cifrado y descifrado. Veamos cómo funciona el algoritmo AES.

¿CÓMO FUNCIONA EL CIFRADO AES?

Entender cómo funciona el algoritmo de cifrado AES es una forma de responder a la pregunta “¿Cuál es la diferencia entre los algoritmos AES y RSA?”.

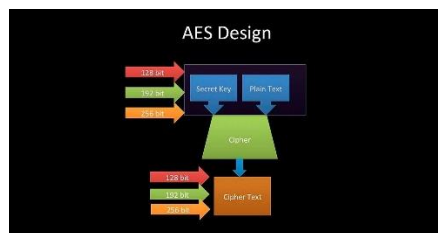


Ilustración 9 Cifrado AES-256 bits, cómo funciona y ¿es realmente seguro?

El cifrado AES es un algoritmo basado en el cifrado simétrico. Se utiliza para cifrar y descifrar y consta de tres componentes clave: la clave de cifrado, el proceso de cifrado y el proceso de descifrado.

- ❖ La clave de cifrado es crucial, ya que ambas partes comunicantes la comparten.

- ❖ Consiste en transformar el texto plano en un formato ilegible denominado texto cifrado.
- ❖ La descriptación toma el texto cifrado y lo transforma de nuevo en texto plano.

En primer lugar, el algoritmo AES amplía la clave secreta para generar un conjunto de claves de ronda. A continuación, AES sustituye cada byte de datos por otro de una tabla de sustitución conocida como S-box. Tras la sustitución, AES baraja los bytes dentro de cada bloque de datos mediante permutación.



Ilustración 10 ¿Qué es el algoritmo AES? | KeepCoding Bootcamps

Esta operación de mezcla se conoce como MixColumns. Combinando estos pasos, el algoritmo de clave simétrica AES garantiza que, aunque los atacantes accedan a los datos cifrados, no podrán descifrarlos sin la clave correspondiente.

¿QUÉ ES RSA?

RSA, siglas de Rivest Shamir y Adleman, es un algoritmo de cifrado asimétrico de clave pública ampliamente utilizado. A diferencia de AES, funciona según el principio de utilizar un par de claves: pública y privada. La clave pública se utiliza para cifrar los datos y la privada para descifrarlos, lo que garantiza la seguridad de la comunicación y la protección de los datos.

Veamos con más detalle cómo funciona el algoritmo de clave asimétrica RSA.



Ilustración 11 RSA

¿CÓMO FUNCIONA EL CIFRADO RSA?

El algoritmo de cifrado RSA se basa en las propiedades matemáticas de los grandes números primos. El proceso comienza generando un par de claves: una clave pública y una clave privada. La clave pública se comparte abiertamente, lo que permite a cualquiera cifrar mensajes para el propietario de la clave privada. Para descifrar el mensaje, el destinatario utiliza su clave privada, que nunca se comparte.



Ilustración 12 Clave RSA: ¿qué es y cómo funciona la criptografía RSA? - IONOS España

La seguridad del algoritmo asimétrico RSA depende de la dificultad de factorizar el producto de dos números primos grandes. Cuanto mayor sea la clave, más intensivo será su cálculo para los atacantes, lo que convierte a RSA en una opción sólida para las comunicaciones seguras.

- ❖ Alice selecciona dos números primos grandes: p y q .
- ❖ Multiplica estos números primos para crear un número grande, n .
- ❖ Alice calcula $\phi(n)$, un número relacionado con sus elecciones de primos.
- ❖ Anuncia públicamente un número concreto.
- ❖ Utilizando fórmulas matemáticas, Alice calcula una clave privada, que llamaremos d .
- ❖ Para enviar un mensaje a Bob, Alice lo convierte en un número utilizando un método específico.
- ❖ Cifra el mensaje utilizando los números públicos (n y el número público elegido).
- ❖ Bob, para leer el mensaje, utiliza la clave privada creada por Alice.

¿QUÉ ES EL ALGORITMO HASH SHA-256?

SHA-256 es un algoritmo hash unidireccional seguro que se utiliza ampliamente como método de cifrado. Pertenece a la familia de algoritmos Secure Hash Algorithm (SHA) que se utilizan para generar valores hash criptográficos. SHA-256 es uno de los algoritmos más populares y utilizados debido a su capacidad para producir un hash de 256 bits.

Un algoritmo hash es una función matemática que convierte datos de entrada en un valor hash de tamaño fijo. SHA-256 es un algoritmo hash criptográfico que se utiliza

para generar un valor hash único e irreversible a partir de cualquier entrada de datos. Es uno de los algoritmos más seguros y confiables para generar valores hash criptográficos.

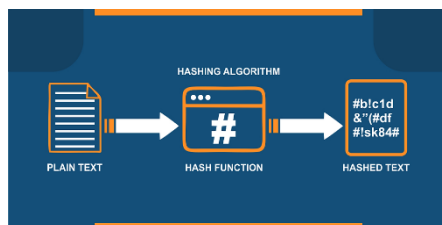


Ilustración 13 SHA-256: The Cryptographic Hash Algorithm Explained

SHA-256 es una evolución de SHA-1, que es un algoritmo hash menos seguro y confiable. SHA-256 es uno de los algoritmos más seguros de la familia SHA y se considera uno de los mejores algoritmos hash disponibles actualmente. Se utiliza en muchas aplicaciones criptográficas, incluidas Bitcoin, Ethereum y otras criptomonedas.

SHA-256 es un algoritmo hash criptográfico que pertenece a la familia Secure Hash Algorithm (SHA) y se utiliza para generar valores hash criptográficos únicos e irreversibles. Es uno de los algoritmos hash más seguros y confiables disponibles actualmente. SHA-256 es uno de los principales algoritmos hash utilizados en criptomonedas y otras aplicaciones criptográficas.

CÓMO FUNCIONA SHA-256

SHA-256 es un algoritmo hash criptográfico que se utiliza para garantizar la integridad de los datos. Es uno de los algoritmos más populares utilizados en criptografía de clave pública y privada. SHA-256 es un algoritmo hash unidireccional, lo que significa que es fácil calcular el hash de una entrada, pero es casi imposible revertir el proceso y descubrir la entrada original.

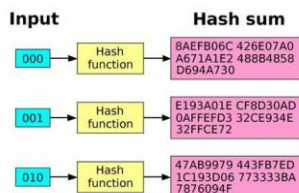


Ilustración 14 Función hash: concepto y aplicación en Bitcoin | Bit2Me Academy

SSL

SSL son las siglas de Secure Sockets Layer, un protocolo diseñado para cifrar y proteger los datos transmitidos entre un servidor web y el navegador de un usuario. En esencia, SSL crea una conexión segura y cifrada entre ambos, garantizando que la información confidencial se mantenga privada y a salvo de miradas indiscretas. SSL es esencial para los sitios web de comercio electrónico y cualquier sitio web que requiera que los usuarios inicien sesión o envíen información confidencial.



Ilustración 15 ¿Qué es el SSL y para qué sirve este certificado

SSL funciona utilizando una clave pública y una clave privada para cifrar y descifrar los datos. La clave pública se utiliza para cifrar los datos y la privada para descifrarlos. Esto garantiza que sólo el destinatario previsto pueda leer los datos.

CÓMO FUNCIONA SSL

Cuando un usuario visita un sitio web que utiliza SSL, su navegador y el servidor web realizan un «apretón de manos» para establecer una conexión segura. Durante este proceso, el servidor web enviará al navegador del usuario una copia de su certificado SSL, que contiene la clave pública necesaria para cifrar los datos.



Ilustración 16 ☐ Cómo funcionan los certificados SSL: guía completa para principiantes

Una vez que el navegador recibe el certificado, comprueba que es válido y ha sido emitido por una autoridad de certificación de confianza. Si todo es correcto, el navegador generará una clave de sesión única y utilizará la clave pública del servidor para cifrarla. La clave de sesión se devuelve al servidor web, que utiliza su clave privada para descifrarla. Ahora ambos pueden comunicarse de forma segura utilizando la clave de sesión.

VENTAJAS DE SSL

Utilizar SSL en su sitio web tiene varias ventajas. En primer lugar, SSL proporciona una conexión segura entre su servidor web y los navegadores de sus usuarios, garantizando que los datos confidenciales se mantienen privados y seguros. Esto es especialmente importante para los sitios web de comercio electrónico y cualquier sitio que requiera que los usuarios inicien sesión o envíen información confidencial.



Ilustración 17 Ventajas de contar con un certificado de seguridad SSL - TrueData Consultore

Otra ventaja del SSL es que ayuda a generar confianza entre los usuarios. Cuando un usuario ve el icono del candado o «https» en la barra de direcciones de su navegador, sabe que el sitio web es legítimo y que su información está segura. Esto puede ayudar a aumentar la confianza del usuario y reducir el riesgo de fraude o robo de identidad.

¿QUÉ ES TLS?

La seguridad de la capa de transporte (TLS) es un protocolo criptográfico que asegura la conexión entre un servidor web y una aplicación web mediante el cifrado de datos. Se aplica a todos los datos que se intercambian en la red, incluidos los correos electrónicos, las sesiones de navegación web y las transferencias de archivos. Como resultado, los hackers no pueden acceder a los datos sensibles de los usuarios, como las credenciales de acceso y los números de las tarjetas de crédito.



Ilustración 18 ¿QUÉ ES TLS? - Grupo ASICA

CÓMO FUNCIONA TLS

Establecido por el Grupo de Trabajo en Ingeniería de Internet (IETF), TLS utiliza el cifrado para que el cliente y el servidor generen una conexión segura entre las aplicaciones. Comienza cuando los usuarios acceden a un sitio web seguro especificando el método de cifrado TLS, como el estándar de cifrado avanzado (AES).

Funciona con dos capas de seguridad: el protocolo de registro TLS y el protocolo de negociación TLS. Estos protocolos utilizan métodos de criptografía simétrica y asimétrica para asegurar la transferencia de datos y las comunicaciones entre los clientes y los servidores web.



Ilustración 19 ¿Qué es TLS y cómo funciona? - Espacios Hosting

El protocolo de enlace TLS, por ejemplo, utiliza la criptografía asimétrica para generar claves públicas y privadas que cifran y descifran los datos. Entonces, el proceso general es el siguiente:

- ❖ El cliente envía una lista de todas las versiones de TLS junto con sugerencias para un conjunto de cifrado y genera un número aleatorio que se utilizará posteriormente.
- ❖ El servidor confirma qué opciones utilizará para iniciar la conexión.
- ❖ El servidor envía un certificado TLS al cliente para el proceso de autenticación.
- ❖ Después de validar el certificado, el cliente crea y envía una clave pre-matriz cifrada por la clave pública del servidor y descifrada por la clave privada del servidor.
- ❖ El cliente y el servidor generan las claves de sesión utilizando los números aleatorios generados previamente y la clave premaster.
- ❖ Tanto el cliente como el servidor tienen un mensaje terminado que ha sido cifrado con una clave de sesión.
- ❖ El proceso de negociación TLS ha finalizado, y tanto el cliente como el servidor han creado un cifrado simétrico seguro.

Además, el protocolo de registro utiliza la encriptación simétrica para generar claves de sesión únicas para cada conexión durante el proceso de negociación. También añade a todos los datos intercambiados un código de autenticación de mensajes basado en hash (HMAC) para verificar la autenticidad de los datos.



Ilustración 20 Qué es TLS y cómo funciona [Guía completa]

Ahora, TLS se está convirtiendo en una práctica estándar para la mayoría de los navegadores modernos y otras aplicaciones, el cual sirve para tres propósitos:

- **Encriptación:** oculta los datos transferidos a terceros mediante información codificada.
- **Autenticación:** TLS garantiza que las identidades de ambas partes son quienes dicen ser mediante un certificado.
- **Integridad:** por último, verifica que los datos transmitidos no han sido falsificados o manipulados durante el proceso de entrega.

¿QUÉ ES EL HTTPS?

Antes de nada, es importante que tengamos claro el significado de HTTP. Es el acrónimo de Hypertext Transfer Protocol (se puede traducir al español como protocolo de transferencia de hiper texto).

HTTPS es lo mismo, con la salvedad de que se le añade el término “seguro”.



Ilustración 21 ¿Qué es el protocolo HTTPS y que ventajas aporta en tu web?

En ambos casos, estamos hablando de un protocolo que define la manera en la que viajan los datos a través de Internet.

- ❖ En el caso de que se utilice el protocolo HTTP, los datos serán accesibles para cualquier persona que pueda captar la comunicación, por lo que no se considera la forma más efectiva de hacerlo.
- ❖ Por su parte, el protocolo HTTPS utiliza una conexión segura ya que se basa en el sistema de cifrado SSL. Esto permite que los datos puedan viajar de

manera segura de un dato a otro, y que no puedan ser captados por cualquiera.



Ilustración 22 Que es el Protocolo HTTPS | Definición y características

Para entender mejor qué es el protocolo HTTPS, es importante saber cómo funciona. A resumidas cuentas, este es su funcionamiento:

- ✚ Abrimos una página de Internet y el navegador intentará conectarse a un sitio que ha sido protegido con SSL.
- ✚ El navegador requerirá que el servidor web se identifique.
- ✚ Será así como el servidor mandará una copia de su certificado SSL a nuestro navegador.
- ✚ El navegador, una vez ha recibido la copia, comprobará si el sitio web es de confianza. Si es así, manda un mensaje que lo acredita.
- ✚ El servidor mandará un acuse de recibo con firma digital que permitirá que se inicie la conexión cifrada.
- ✚ Así es como empezarán a transferirse datos cifrados entre el servidor y el navegador.

¿PARA QUÉ SIRVE EL PROTOCOLO HTTPS?



Ilustración 23 ¿Qué es el protocolo HTTPS y para qué sirve?

Para que puedas conocer con más detalle qué es HTTPS en informática, aquí tienes un pequeño listado con sus utilidades principales:

- Creación de un canal encriptado

El HTTPS permite dar forma a un canal encriptado para trabajar con aquella información que es más sensible.

Eso sí, habrá que tener en cuenta que el nivel de cifrado estará supeditado al navegador que se utilice y el servidor remoto con el que se está interactuando.

- No hará falta usar ningún software adicional

Al usar el protocolo no será necesario instalar ningún tipo de software o funcionalidad adicional. Por ello, será utilizable por cualquier tipo de navegante sin ninguna restricción.

Tal y como veremos a continuación cuando hablemos de las ventajas del HTTPS, esto es muy interesante, ya que ayuda a fomentar la confianza en el cliente, elevando las posibilidades de que se produzca la conversión deseada.

- Total, integración

El protocolo HTTPS ser compatible sin ningún tipo de problema con la mayoría de los navegadores web actuales, como:

- Mozilla Firefox
- Google Chrome
- Microsoft Edge
- Opera
- Safari
- Identificación inequívoca de que la web es segura

Lo más habitual es que aparezca un rasgo distintivo en el navegador que nos indique que la página es segura. Suele ser un pequeño candado que se sitúa en la zona izquierda en la barra de direcciones.

Identificación inequívoca de que la web es segura.



Ilustración 24 Http y Https: ¿Qué son y para qué sirven? - 1456 Blog

Además, el término HTTPS aparecerá en la dirección URL, elevando todavía más la confianza del usuario.

- No almacena contenido en caché

La información que se transmite a través del protocolo HTTPS no almacena ningún tipo de información en caché. Sin embargo, esto podría ser una ventaja o un inconveniente, dependiendo de la situación.

El hecho de que no se almacene en caché es interesante porque no quedan datos residuales a los que se pueda acceder para intentar vulnerarlos. A cambio, la página tardará más tiempo en cargarse (cuestión de milisegundos).

- Proceso de verificación complejo

En el caso de que una persona no autorizada consiga obtener los datos que han sido transmitidos a través de este protocolo, no será capaz de descifrar la información debido a que le llegará encriptada.

Tan solo le llegará descriptada si se han concluido todos los datos que indicamos en los apartados anteriores.

SEGURIDAD

Esta es la principal diferencia entre ambos protocolos,

HTTPS es la versión segura de HTTP, ya que cifra la información enviada y recibida para protegerla contra ataques externos.

Utiliza un protocolo SSL/TLS para crear un canal cifrado, algo vital para proteger información sensible.



Ilustración 25 seguridad-https - Creamos tu video para empresas

HTTPS es el estándar que debe tener cualquier sitio que manipule información delicada como datos bancarios, información personal y credenciales de inicio de sesión.

de hecho, navegadores como Chrome o Firefox marcan las páginas HTTP como “no seguras” para alertar a los usuarios sobre los potenciales riesgos en su seguridad.

CIFRADO

HTTP transmite datos en texto plano. Esto quiere decir que puede ser interceptado y leído por cualquier parte que capture los datos durante su tránsito.

Sin embargo, HTTPS cifra los datos. Emplea certificados SSL/TLS.



Ilustración 26 HTTPS: Cifrado en la web | AEPD

Por tanto, aunque los datos sean interceptados, no pueden ser leídos sin la clave de cifrado correcta

SFTP

Un protocolo seguro de transferencia de archivos es un protocolo de transferencia de archivos basado en web para archivos grandes. Está basado en FTP y contiene componentes de seguridad Secure Shell (SSH). Secure Shell es un componente de criptografía de seguridad de Internet que promueve transferencias seguras.



Ilustración 27 Protocolo SFTP: o que é e para que serve

¿Cómo funciona SFTP?

SFTP utiliza un flujo de datos de shell seguro. Establece una conexión segura y luego protege los datos mientras se transfieren. Este proceso utiliza varios cifrados para ayudar en el transporte de datos y garantizar que los archivos y comandos que se procesan no se lean.

Las claves SSH deben generarse con anticipación y ayudan a evitar que usuarios fraudulentos se conecten al servidor. Opera en un modelo cliente-servidor, mientras que los datos normalmente están presentes en el servidor. Aunque el servidor esté ubicado en otro lugar, el cliente puede acceder fácilmente a los datos mediante la emisión de una solicitud. Por ejemplo, cuando un usuario selecciona un archivo, la solicitud viaja por la red hasta llegar al servidor.



Ilustración 28 Aprende qué es SFTP y cómo funciona

Esta información se envía luego al servidor que la ha solicitado. El usuario finalmente recibirá el archivo y realizará las modificaciones necesarias. El intercambio de archivos está protegido ya que todos los archivos se transfieren de forma cifrada utilizando el protocolo SFTP. Las claves SSH ayudan a transferir la clave pública a cualquier sistema para acceder.

¿QUÉ ES SSH?

SSH o Secure Shell, es un protocolo de administración remota que le permite a los usuarios controlar y modificar sus servidores remotos a través de Internet a través de un mecanismo de autenticación.

Proporciona un mecanismo para autenticar un usuario remoto, transferir entradas desde el cliente al host y retransmitir la salida de vuelta al cliente. El servicio se creó como un reemplazo seguro para el Telnet sin cifrar y utiliza técnicas criptográficas para garantizar que todas las comunicaciones hacia y desde el servidor remoto sucedan de manera encriptada.



Ilustración 29 ¿Qué es SSH?

La imagen de abajo muestra una ventana típica de SSH. Cualquier usuario de Linux o macOS puede usar SSH en su servidor remoto directamente desde la ventana del terminal. Los usuarios de Windows pueden aprovechar los clientes SSH como Putty. Puedes ejecutar comandos shell de la misma manera que lo harías si estuvieras operando físicamente el equipo remoto.

¿Cómo funciona SSH?

Si usas Linux o Mac, entonces usar el protocolo SSH es muy fácil. Si utilizas Windows, deberás utilizar un cliente SSH para abrir conexiones SSH. El cliente SSH más popular es PuTTY, puedes aprender más acerca de él aquí.

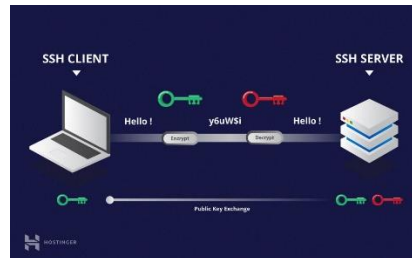


Ilustración 30 ¿Cómo funciona el SSH? (2023)

Para usuarios de Mac y Linux, dirígete a tu programa de terminal y sigue este procedimiento:

El comando SSH consta de 3 partes distintas:

`ssh {user}@{host}`

CONCLUSIÓN

Como conclusión la seguridad de la información es esencial en el mundo digital, y comprender cómo proteger datos sensibles es crucial. A través de programas en Java, podemos validar y generar contraseñas seguras que cumplan con criterios de longitud, complejidad, y ausencia de caracteres repetidos o espacios. Además, el uso de certificados SSH y SSL refuerza la autenticidad y confidencialidad en las comunicaciones, asegurando que solo usuarios autorizados accedan a sistemas y que las conexiones web sean seguras. La investigación sobre conceptos clave como cifrado, firmas digitales, y algoritmos como AES, RSA y SHA-256, junto con los estándares y protocolos de seguridad como SSL, TLS, HTTPS, SFTP y SSH, proporciona una base sólida para entender y aplicar prácticas de protección de la información, garantizando la seguridad en diversas plataformas tecnológicas.

Bibliografía

<https://portalcripto.com.br/es/DICIONARIO/Algoritmo-hash-sha-256-qu%C3%A9-es-y-c%C3%B3mo-funciona/#:~:text=SHA-256%20es%20un%20algoritmo%20hash%20unidireccional%20seguro%20que,ca pacidad%20para%20producir%20un%20hash%20de%20256%20bits.>

<https://www.ssldragon.com/es/blog/rsa-aes-cifrado/>

<https://www.hostinger.mx/tutoriales/que-es-tls>

<https://lovtechnology.com/que-es-ssl-secure-sockets-layer-como-funciona-y-para-que-sirve/>

<https://www.siteground.es/blog/que-es-https-y-para-que-sirve-guia-completa/>

<https://tecno-simple.com/que-es-sftp-protocolo-seguro-de-transferencia-de-archivos/#:~:text=El%20Protocolo%20seguro%20de%20transferencia%20de%20a rchivos%2C%20tambi%C3%A9n,viajan%20entre%20sistemas%20utilizando%20AES%20y%20otras%20t%C3%A9cnicas.>

<https://www.hostinger.mx/tutoriales/que-es-ssh>