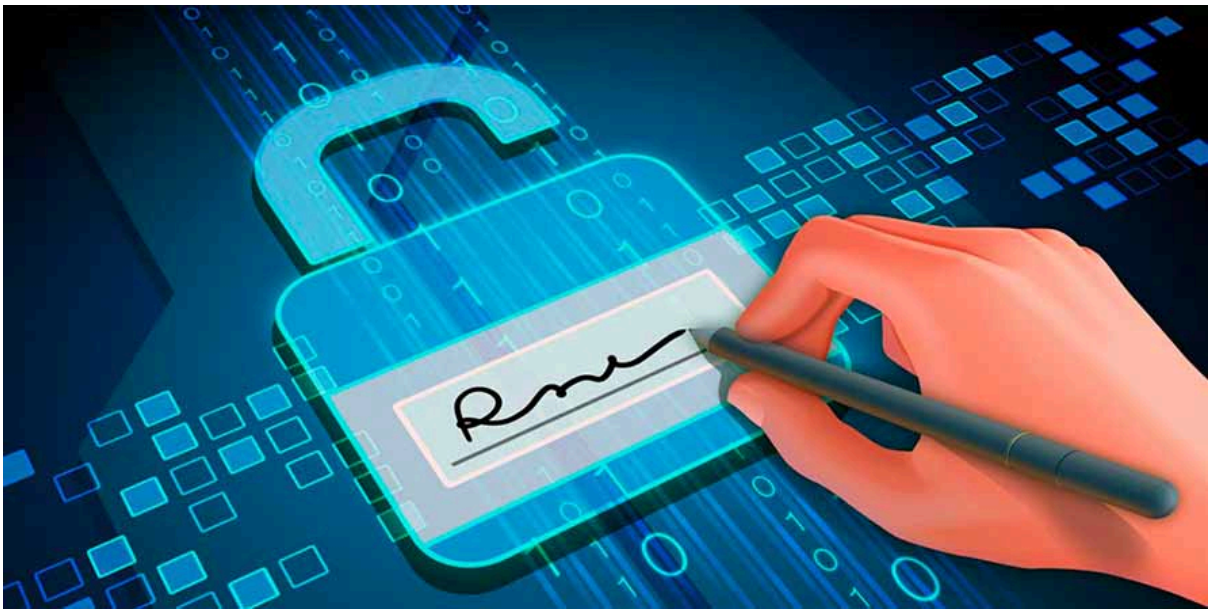
Firma Digital



Izan Christen

```
KeyPair parClaves = generarParDeClaves();
PrivateKey clavePrivada = parClaves.getPrivate();
PublicKey clavePublica = parClaves.getPublic();
System.out.println("Par de claves RSA generado correctamente.\n");
```

Genero las claves, tanto probada como publica.

```
Scanner sc = new Scanner(System.in);
System.out.print("Introduce el mensaje a firmar: ");
String mensaje = sc.nextLine();
System.out.println();
```

Pido input de mensaje a cifrar.

```
byte[] hash = calcularHash(mensaje);
System.out.println("Hash SHA-256 del mensaje: " + bytesAHex(hash) + "\n");
```

Calculo el Hash con el metodo calcularHash

```
byte[] firma = firmarMensaje(mensaje, clavePrivada);
System.out.println("Firma digital (Base64): " + Base64.getEncoder().encodeToString(firma) + "\n");
```

Firmo el mensaje con la clave privada

```
boolean validaOriginal = verificarFirma(mensaje, firma, clavePublica);
if (validaOriginal) {
    System.out.println("Verificación con mensaje original: VALIDA\n");
} else {
    System.out.println("Verificación con mensaje original: INVALIDA\n");
}
```

Válido la firma con la clave pública.

```
String mensajeAlterado = mensaje + " (modificado)";
boolean validaAlterado = verificarFirma(mensajeAlterado, firma, clavePublica);
System.out.println("Mensaje modificado: " + mensajeAlterado);
if (validaAlterado) {
    System.out.println("Verificación con mensaje alterado: VALIDA (ERROR)\n");
} else {
    System.out.println("Verificación con mensaje alterado: INVALIDA\n");
}
```

Vuelo a validar pero con un mensaje modificado.

```
} catch (Exception e) {
    System.err.println("Se produjo un error: " + e.getMessage());
    e.printStackTrace();
}
```

Control de errores

```
private static KeyPair generarParDeClaves() throws NoSuchAlgorithmException {  1 usage  IzanChristen
    KeyPairGenerator generador = KeyPairGenerator.getInstance( algorithm: "RSA");
    generador.initialize( keysize: 2048);
    return generador.generateKeyPair();
}

private static byte[] calcularHash(String mensaje) throws NoSuchAlgorithmException {  1 usage  IzanChristen
    MessageDigest md = MessageDigest.getInstance( algorithm: "SHA-256");
    return md.digest(mensaje.getBytes());
}

private static byte[] firmarMensaje(String mensaje, PrivateKey clavePrivada) throws Exception {  1 usage  IzanChristen
    Signature firma = Signature.getInstance( algorithm: "SHA256withRSA");
    firma.initSign(clavePrivada);
    firma.update(mensaje.getBytes());
    return firma.sign();
}

private static boolean verificarFirma(String mensaje, byte[] firmaBytes, PublicKey clavePublica) throws Exception {
    Signature verificador = Signature.getInstance( algorithm: "SHA256withRSA");
    verificador.initVerify(clavePublica);
    verificador.update(mensaje.getBytes());
    return verificador.verify(firmaBytes);
}

private static String bytesAHex(byte[] bytes) {  1 usage  IzanChristen
    StringBuilder sb = new StringBuilder();
    for (byte b : bytes) {
        sb.append(String.format("%02x", b));
    }
    return sb.toString();
}
```

Métodos que utilizo anteriormente para:

- Generar las claves
- Calcular el Hash
- Firmar el mensaje
- Verificar la firma
- Pasar de Bytes a Hex