

# ***Desarrollo de software seguro***



**ceti** **CENTRO DE ENSEÑANZA  
TÉCNICA INDUSTRIAL**

***Profesor:*** José Luis García Cerpas.

***Nombre:*** Henry Daniel Hernández Morales.

***Registro:*** 21110385.

***Grupo:*** 7°N.

***Carrera:*** Ingeniería en Desarrollo de Software.

***Tarea:*** Algoritmos criptográficos.

## **Criptografía simétrica.**

En el cifrado simétrico, sólo hay una clave, y todas las partes implicadas utilizan la misma clave para cifrar y descifrar la información. Al utilizar una única clave, el proceso es sencillo, como en el siguiente ejemplo: encriptas un correo electrónico con una clave única, envías ese correo a algún amigo, y esa persona utilizará la misma clave simétrica para desbloquear/desencriptar el correo.

Las ventajas de este tipo de cifrado son su mayor rendimiento y su bajo consumo de recursos, pero es intrínsecamente más antiguo y menos seguro que su homólogo. La razón es sencilla: si se usa una única clave para toda la empresa, significa que se está confiando en una clave que tendremos que compartir mucho.

Por esta razón, el cifrado simétrico es estupendo cuando se trabaja con datos sensibles a gran escala, o en tareas de cifrado que pretenden ocultar permanentemente la información sin necesidad de descifrarla. Por ejemplo, cuando se activa BitLocker en un ordenador con Windows para cifrar todos los discos duros. Al desbloquear el PC con su código de acceso, el usuario descifrá los datos sin riesgo de exponer su clave secreta de cifrado. Otro ejemplo son las VPN, que cifran su tráfico de red con una clave local y no tienen la necesidad de compartirla fuera de su propio uso.

### **AES (Advanced Encryption Standard)**

- **Descripción y funcionamiento:** AES es un algoritmo de cifrado por bloques con longitudes de clave de 128, 192 o 256 bits. Utiliza sustituciones, permutaciones y operaciones matemáticas sobre matrices.
- **Fortalezas:** Alta seguridad, eficiente en software y hardware.
- **Debilidades:** Vulnerable a ataques de fuerza bruta si se usa una clave débil.
- **Casos de uso:** Cifrado de datos en almacenamiento y transmisión segura de información.
- **Nivel de seguridad:** Seguro según estándares actuales.

### **DES (Data Encryption Standard)**

- **Descripción y funcionamiento:** DES utiliza una clave de 56 bits y opera en bloques de 64 bits con 16 rondas de permutaciones y sustituciones.
- **Fortalezas:** Fue ampliamente adoptado y probado.
- **Debilidades:** Clave demasiado corta, vulnerable a ataques de fuerza bruta.
- **Casos de uso:** Usado históricamente en sistemas bancarios.
- **Nivel de seguridad:** Obsoleto.

### 3DES (Triple DES)

- **Descripción y funcionamiento:** Aplica DES tres veces con diferentes claves para aumentar la seguridad.
- **Fortalezas:** Más seguro que DES.
- **Debilidades:** Ineficiente en comparación con AES, clave efectiva de solo 112 bits.
- **Casos de uso:** Seguridad en transacciones financieras.
- **Nivel de seguridad:** En desuso, recomendado reemplazo por AES.

### Criptografía asimétrica.

Se basa en una codificación de información basada en dos claves: una privada y una pública. De esta manera, el remitente conserva la clave privada y la pública puede entregarse a cualquier receptor. La Clave privada permite descifrar todos los mensajes cifrados con la clave pública; con la pública solo podemos descifrar los mensajes cifrados con la clave privada original. Abordemos mejor el tema de las claves:

- **Claves Públicas:** La clave pública puede encriptar mensajes que sólo se descifran con la clave privada. Esto significa que nadie con la clave pública puede descifrar ese mensaje. Por esto surge la confidencialidad, ya que solo el receptor, podrá interpretar el mensaje.
- **Claves Privadas:** Con la clave privada podemos cifrar información mientras la persona posea el par de la clave pública. Este proceso no brinda la misma confidencialidad, ya que cualquiera con la clave pública podría leer el mensaje, pero les otorga autenticidad a los mensajes. Esto se debe a que solo el que tiene la clave privada puede cifrar la información de la forma en que solo el que tiene la clave pública la puede descifrar.

### RSA (Rivest-Shamir-Adleman):

- Es uno de los algoritmos de cifrado asimétrico más antiguos y ampliamente utilizados. Se basa en la dificultad de factorizar grandes números primos. Además, la seguridad de RSA aumenta con el tamaño de la clave (típicamente 2048 bits o más).
- Usos comunes: Firmas digitales, cifrado de datos sensibles, y protección de comunicaciones en internet (SSL/TLS).

### ECC (Elliptic Curve Cryptography):

- Utiliza las propiedades matemáticas de las curvas elípticas para proporcionar seguridad. Ofrece una seguridad comparable a RSA pero con claves mucho

más pequeñas. Proporciona un nivel de seguridad similar a claves de RSA mucho más grandes.

- Usos comunes: SSL/TLS, criptomonedas (como Bitcoin), y aplicaciones móviles debido a su eficiencia.

### **DSA (Digital Signature Algorithm):**

- Es un estándar de firma digital adoptado por el Instituto Nacional de Estándares y Tecnología (NIST) de EE.UU. Se basa en el problema del logaritmo discreto. Proporciona integridad y autenticidad a través de firmas digitales.
- Usos comunes: Autenticación y verificación de documentos electrónicos y software.

### **¿Qué es el MD5?**

El MD5 (algoritmo de resumen de mensajes) es un protocolo criptográfico que se usa para autenticar mensajes y verificar el contenido y las firmas digitales. El MD5 se basa en una función hash que verifica que un archivo que ha enviado coincide con el que ha recibido la persona a la que se lo ha enviado. Anteriormente, MD5 se usaba para el cifrado de datos, pero ahora se utiliza principalmente para la autenticación.

### **¿Cómo funciona el MD5?**

El MD5 ejecuta archivos enteros a través de un algoritmo de hashing matemático para generar una firma que pueda compararse con un archivo original. Así se puede verificar que un archivo recibido coincida con el archivo original que se envió, lo que garantiza que los archivos correctos lleguen a su destino.

El algoritmo de hashing MD5 convierte los datos en una cadena de 32 caracteres. Por ejemplo, la palabra «frog» siempre genera este hash: 938c2cc0dcc05f2b68c4287040cfcf71. Del mismo modo, un archivo de 1,2 GB también genera un hash con el mismo número de caracteres. Cuando le envía ese archivo a alguien, el ordenador verifica su hash para asegurarse de que coincida con el que nosotros enviamos.

Si se cambia un solo bit en un archivo, independientemente de lo grande que sea el archivo, la información del hash cambiará completa e irreversiblemente. Nada, salvo una copia exacta, pasará la prueba MD5.

### **¿Para qué se utiliza el MD5?**

El MD5 se usa principalmente para verificar archivos. Es mucho más sencillo usar el hash MD5 para cotejar una copia de un archivo con su original que comprobar bit por bit si ambas copias coinciden.

El MD5 se usaba antes para la seguridad y el cifrado de datos, pero hoy en día su uso principal es la autenticación. Dado que un hacker puede crear un archivo que tenga exactamente el mismo hash que otro totalmente diferente, el MD5 no es seguro en caso de que alguien manipule un archivo. Pero si tan solo está copiando un archivo de un lugar a otro, el MD5 nos servirá.

Puesto que el MD5 ya no se usa con fines de cifrado, si se necesita asegurar nuestros archivos, es buena idea considerar la posibilidad de instalar el mejor software de cifrado que pueda encontrar o aprender a activar el cifrado del Wi-Fi en la configuración del router.

### SHA-256.

En criptografía, se conoce como SHA-256 al **Algoritmo de Hash Seguro** (*Secure Hash Algorithm*) de 256 bits, que se utiliza para la seguridad criptográfica. Estos algoritmos generan hashes (cadenas de caracteres de longitud fija) irreversibles y únicos. Cuanto mayor sea la cantidad de hashes posibles, menor será la probabilidad de que dos valores creen el mismo hash.

Ejemplos:

- SHA2-256("kaixo") =  
d050c1cbf2956cde459da5c8fd8851da352c10c52f758d0e3bfbef2473a802b7
- SHA2-256("agur") =  
d3404c774cf3207c4fe9d82751da0b96a7db8679b11e71cb7fc4cd4e40ee247c

La familia SHA (Algoritmo de Hash Seguro o *Secure Hash Algorithm*) es un sistema de funciones hash criptográficas desarrollado por la Agencia Nacional de Seguridad estadounidense y publicado por el *National Institute of Standards and Technology* (NIST). La primera de estas funciones fue publicada en 1993 bajo el título *Secure Hash Standard*, FIPS PUB 180, y se conoce oficialmente como SHA. Esta función se ha popularizado como SHA-0, para evitar confusiones con sus sucesoras. Dos años después del desarrollo de la primera función fue publicado el primer sucesor de SHA, que recibió el nombre de SHA-1.

Desde entonces se han publicado cuatro variantes más bajo el nombre de SHA-2; las diferencias entre ellas se basan en un diseño algo modificado y en rangos de salida incrementados: se trata de SHA-224, **SHA-256**, SHA-384, y SHA-512. SHA-3 es el último miembro de la familia, publicado por NIST en agosto de 2015; su algoritmo genera hashes de la misma longitud que SHA-2 mediante la utilización de un método diferente basado en el algoritmo Keccak.

El procesamiento del algoritmo SHA-256 consta de estos 5 pasos:

1. **Se incorporan bits adicionales de relleno al mensaje de entrada.** Este relleno sirve para ocultar la estructura y longitud del contenido, dificultando al ciberdelincuente su posible ataque. Esta operación consiste en la adición de un "1" seguido de los "0" que sean necesarios hasta completar el bloque de 512 bits.
2. **Se le añade un bloque de 64 bits** que represente la longitud del mensaje original antes de ser relleno.
3. **Se inicializa una caché de 256 bits** para almacenar los resultados intermedios y finales de la función hash. Consta de 8 registros (A, B, C, D, E, F, G, H) de 32 bits cada uno.
4. **Se procesa el mensaje en bloques de 512 bits.** El algoritmo utiliza seis funciones lógicas básicas y consta de 64 operaciones iterativas. Cada paso toma el valor del búfer de 256 bits ABCDEFGH como entrada y luego actualiza el contenido del búfer. Cada paso utiliza un valor constante Kt de 32 bits y un Wt de 32 bits.
5. Una vez procesados todos los bloques de 512 bits, la salida obtenida por el último bloque generado por el algoritmo SHA-256 es un **resumen de 256 bits**.

El algoritmo SHA-256 es utilizado en un gran número de herramientas de seguridad y protocolos, entre ellos TLS, SSL, PGP, SSH o IPsec. Otra aplicación de interés que brinda SHA-256 está relacionada con el bitcoin y otras criptomonedas. El protocolo de Prueba de Trabajo (del inglés Proof-Of-Work) fue diseñado teniendo en cuenta este algoritmo, requiriendo resolver complejos acertijos criptográficos en los que debía hacerse un uso intensivo de esta función.

En la actualidad, SHA-256 se considera seguro, a pesar de tener las mismas debilidades matemáticas que su predecesor SHA-1, y sabiendo que los mejores ataques públicos han conseguido romper 46 de las 64 iteraciones. Aun así, existen sucesores, como SHA3, que pueden ser utilizados como reemplazo de SHA-256.

### Ejemplo práctico con Python implementando el cifrado AES.

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes
from cryptography.hazmat.primitives import padding
import os

# Generar clave y vector de inicialización
clave = os.urandom(32)
iv = os.urandom(16)

def cifrar(mensaje):
    padder = padding.PKCS7(128).padder()
    mensaje_padded = padder.update(mensaje.encode()) + padder.finalize()
```

```

cipher = Cipher(algorithms.AES(clave), modes.CBC(iv))
encryptor = cipher.encryptor()
return encryptor.update(mensaje_padded) + encryptor.finalize()

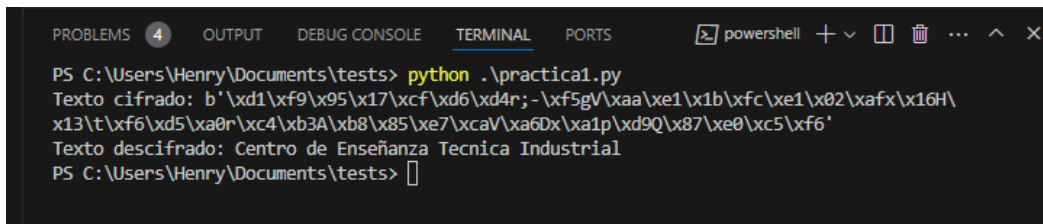
def descifrar(texto_cifrado):
    cipher = Cipher(algorithms.AES(clave), modes.CBC(iv))
    decryptor = cipher.decryptor()
    mensaje_padded = decryptor.update(texto_cifrado) +
decryptor.finalize()
    unpadder = padding.PKCS7(128).unpadder()
    return unpadder.update(mensaje_padded) + unpadder.finalize()

mensaje = "Centro de Enseñanza Tecnica Industrial"
cifrado = cifrar(mensaje)
descifrado = descifrar(cifrado)

print("Texto cifrado:", cifrado)
print("Texto descifrado:", descifrado.decode())

```

Ejecución:



```

PS C:\Users\Henry\Documents\tests> python .\practica1.py
Texto cifrado: b'\xd1\xf9\x95\x17\xcf\xd6\xd4r;- \xf5gV\xaa\xe1\x1b\xfc\xe1\x02\xaf\x16H\x13\t\xf6\xd5\xa0r\xc4\xb3A\xb8\x85\xe7\xcaV\xa6Dx\xa1p\xd9Q\x87\xe0\xc5\xf6'
Texto descifrado: Centro de Enseñanza Tecnica Industrial
PS C:\Users\Henry\Documents\tests>

```

## Conclusión.

Se puede entender la importancia de la criptografía en la seguridad de la información gracias a esta investigación. La implementación de AES en Python ha demostrado su efectividad para proteger datos. Es de suma importancia escoger el algoritmo que nos parezca el más adecuado según las necesidades del sistema.

## Referencias bibliográficas.

- SHA-256. (s. f.). Cyberzaintza.  
<https://www.ciberseguridad.eus/ciberglosario/sha-256>
- Freda, A. (2023, 23 febrero). ¿Qué es el algoritmo de hashing MD5 y cómo funciona? ¿Qué Es el Algoritmo de Hashing MD5 y Cómo Funciona?  
<https://www.avast.com/es-es/c-md5-hashing-algorithm>
- Cifrado simétrico y asimétrico: guía completa sobre criptografía. (s. f.).  
<https://preyproject.com/es/blog/tipos-de-cifrado-simetrico-o-asimetrico-rsa-o-aes#:~:text=3DES%20o%20Est%C3%A1ndar%20de%20Cifrado%20de%20Datos%20Triple&text=Este%20algoritmo%20sim%C3%A9trico%20es%20una,en%20una%20de%20168%20bits.>

- Montenegro, I. (2021, 27 julio). Encriptación Simétrica y Asimétrica: Conoce sus diferencias. GB Advisors. <https://www.gb-advisors.com/es/encriptacion-simetrica-y-asimetrica-conoce-sus-diferencias/#:~:text=Sim%C3%A9trica%20vs%20Asim%C3%A9trica%3A%20Diferencias&text=Los%20algoritmos%20de%20encriptaci%C3%B3n%20sim%C3%A9trica,y%20otra%20distinta%20para%20desencriptarlos.>
- Chat GPT