



HASHES

Descripción breve

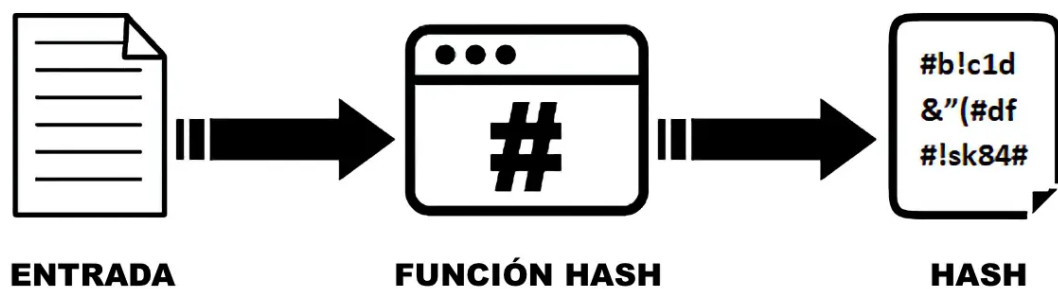
Este trabajo explora las funciones hash , herramientas esenciales en ciberseguridad para garantizar la integridad, autenticación y protección de datos. Se analizan sus características principales, como la unidireccionalidad, resistencia a colisiones y aplicaciones prácticas en almacenamiento seguro de contraseñas, blockchain y verificación de integridad. Además, se comparan diferentes algoritmos hash (MD5, SHA-1, SHA-256, BLAKE2, Argon2) y se discuten sus fortalezas y debilidades.

Beatriz Suarez y Leonardo Duarte

Funciones de Hash

Introducción

Las funciones hash son pilares fundamentales de la criptografía moderna y desempeñan un papel crucial en la seguridad de sistemas digitales. Estas funciones transforman datos de cualquier tamaño en una cadena de longitud fija, conocida como "hash", que actúa como una huella digital única. Además de su uso en criptografía, las funciones hash son ampliamente utilizadas en aplicaciones como verificación de integridad, almacenamiento seguro de contraseñas y blockchain. En este trabajo, exploraremos su funcionamiento, características, aplicaciones prácticas, desafíos asociados y su evolución en el contexto de la ciberseguridad.



1. Funcionamiento y Tipos de Hash

1.1 Definición

Una función hash es una función matemática que toma un conjunto de datos de entrada (de cualquier tamaño) y produce una salida de longitud fija. Esta salida se denomina "hash" o "resumen". Las funciones hash son únicas, deterministas y unidireccionales. Actúan como una huella digital digital que representa los datos originales.

Visualización del Proceso:

```
C: > Users > beal > Downloads > ☰ proceso
1  Entrada (datos de cualquier tamaño) → Función Hash → Salida (hash de longitud fija)
2  Ejemplo:
3  Entrada: "Hola, mundo!"
4  Hash SHA-256: 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824
```

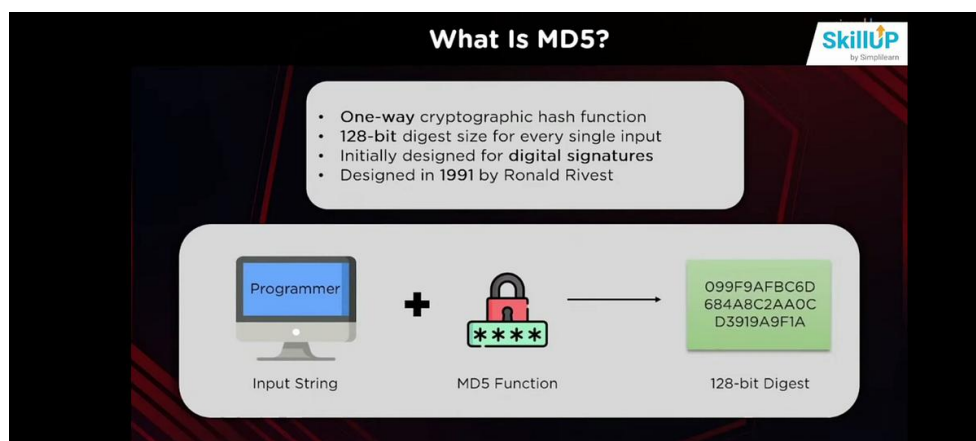
1.2 Características Principales

- Unidireccionalidad : Es computacionalmente imposible recuperar los datos originales a partir del hash.
- Determinista : Para la misma entrada, siempre se genera el mismo hash.
- Sensibilidad a cambios : Un cambio mínimo en la entrada produce un hash completamente diferente (efecto avalancha).
- Rapidez : Las funciones hash deben ser rápidas de calcular para ser prácticas.
- Resistencia a colisiones : Deben minimizar la probabilidad de que dos entradas diferentes produzcan el mismo hash.

1.3 Tipos Comunes de Funciones Hash

1. MD5 (128 bits):

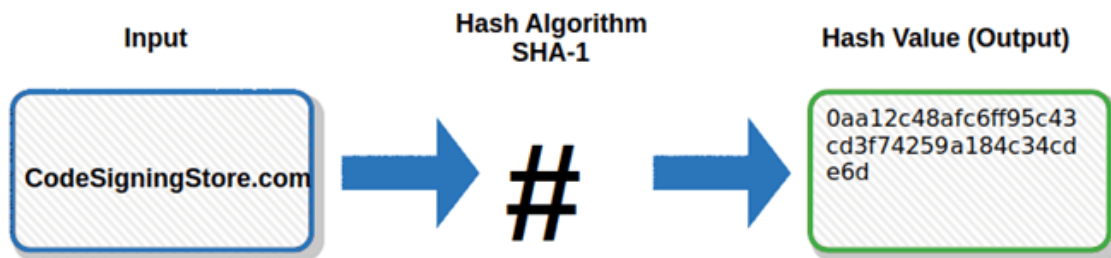
- Fue ampliamente utilizado en el pasado debido a su rapidez, pero hoy en día está considerado inseguro debido a su vulnerabilidad a ataques de colisión.
- Ejemplo de hash MD5: **5d41402abc4b2a76b9719d911017c592.**



2. SHA-1 (160 bits):

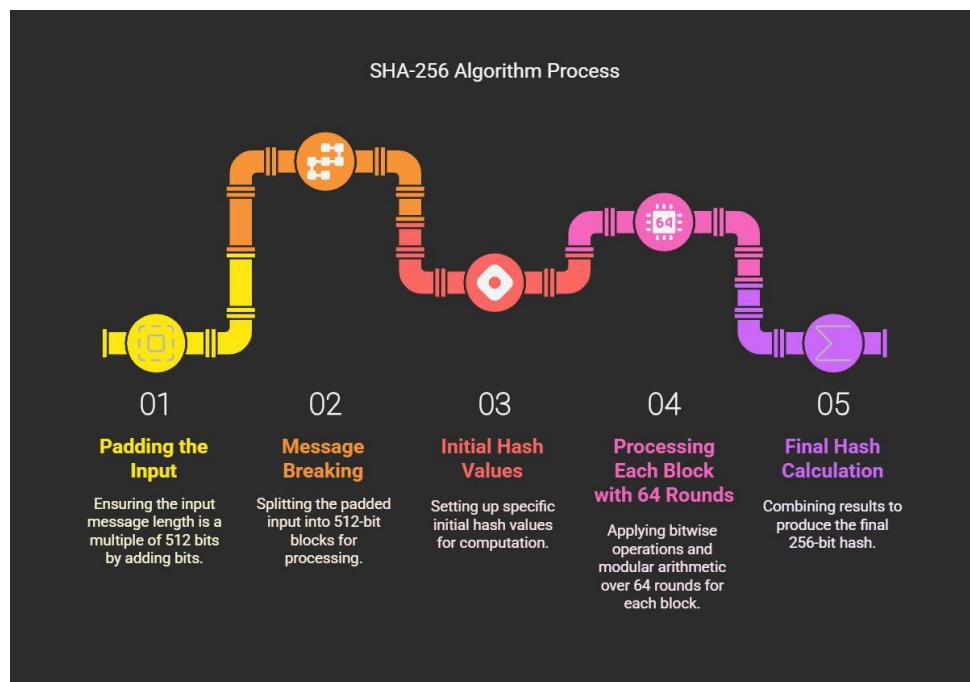
- Introducido por la NSA como una mejora sobre MD5, también ha sido declarado obsoleto debido a vulnerabilidades detectadas.
- Ejemplo de hash SHA-1:
aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d.

SHA-1 Hashing Algorithm for CodeSigningStore.com



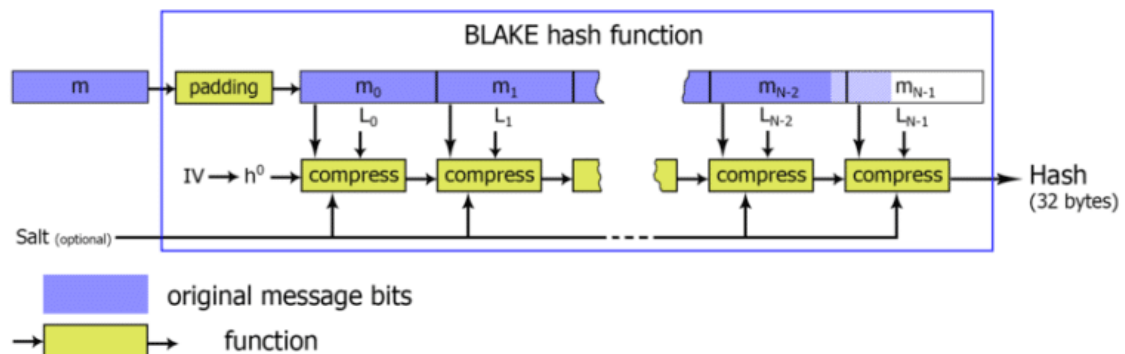
3. SHA-256 (256 bits):

- Parte de la familia SHA-2, ofrece un alto nivel de seguridad y es ampliamente utilizado en aplicaciones modernas como certificados SSL/TLS y blockchain.
- Ejemplo de hash SHA-256:
2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824.



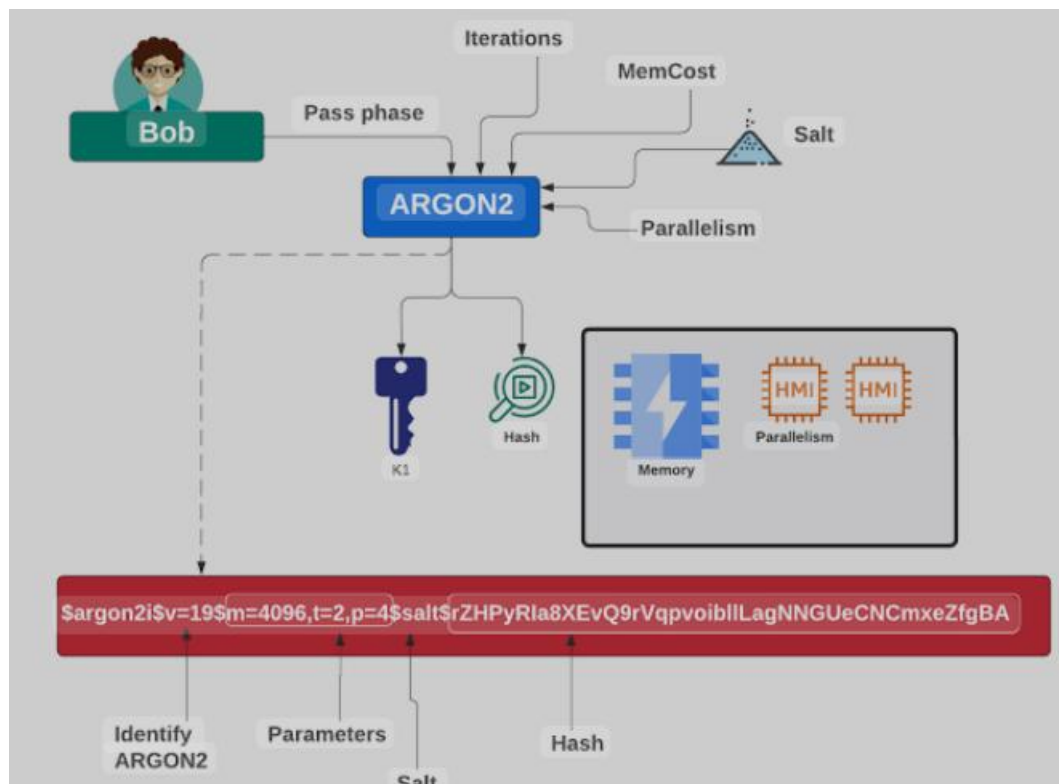
4. BLAKE2:

- Una alternativa moderna a SHA-2 y SHA-3, conocida por su alta velocidad y seguridad.
- Ejemplo de hash BLAKE2:
1a79a4d60de6718e8e5b326e338ae53304b002c5c45bb799c9ec92a81a770b66.



5. Argon2:

- Diseñado específicamente para derivación de claves y almacenamiento seguro de contraseñas.
- Ejemplo de hash Argon2:
\$argon2id\$v=19\$m=65536,t=3,p=4\$c2FsdA\$hashedpassword.



1.4 Proceso de Generación de Hashes

El proceso de generación de hashes se divide en varias etapas:

1. Preprocesamiento :

- El mensaje se divide en bloques de tamaño fijo (por ejemplo, 512 bits para SHA-256).
- Si el último bloque no está completo, se añade padding (relleno) para completarlo.

2. Inicialización :

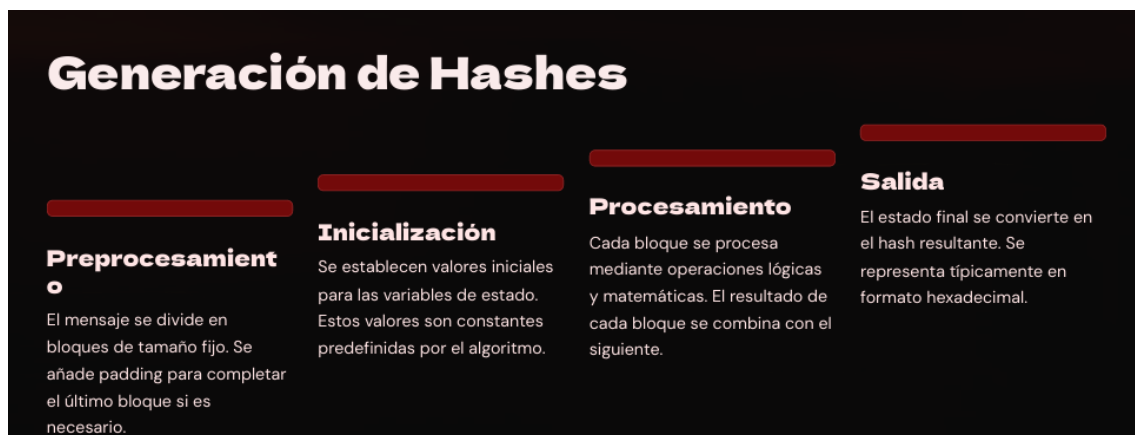
- Se establecen valores iniciales para las variables de estado. Estos valores son constantes predefinidas por el algoritmo.

3. Procesamiento :

- Cada bloque se procesa mediante operaciones lógicas (AND, OR, XOR) y matemáticas (rotaciones, sumas modulares).
- El resultado de cada bloque se combina con el siguiente mediante funciones de compresión.

4. Salida :

- El estado final se convierte en el hash resultante, representado típicamente en formato hexadecimal.



2. Colisiones en Funciones Hash

2.1 ¿Qué es una colisión?

Una colisión ocurre cuando dos entradas diferentes generan el mismo valor hash. Según el principio del palomar, si hay más entradas posibles que valores hash únicos, eventualmente habrá **colisiones**.

Ejemplo Histórico: Ataque a SHA-1 En 2017, investigadores demostraron una colisión práctica en SHA-1 mediante el proyecto SHAttered . Crearon dos archivos PDF diferentes con el mismo hash SHA-1, demostrando que el algoritmo ya no era seguro para aplicaciones críticas.

2.2 Impacto de las Colisiones

1. Autenticación : Un atacante podría crear un documento malicioso con el mismo hash que uno legítimo, lo que comprometería la autenticidad del sistema.
2. Almacenamiento de contraseñas : Si un atacante encuentra una colisión, podría suplantar la identidad de un usuario.
3. Blockchain : Una colisión podría permitir transacciones fraudulentas o manipulación del registro inmutable.

3. Usos en Seguridad Informática

3.1 Verificación de Integridad

Las funciones hash se utilizan para generar checksums que verifican que archivos o mensajes no han sido modificados durante la transmisión.

Ejemplo Práctico:

```
C: > Users > beasl > Downloads > ⚙ proceso
1 Archivo descargado: setup.exe
2 Hash SHA-256 proporcionado: abc123...
3 Hash calculado: abc123... (coinciden, el archivo no ha sido alterado)
```

3.2 Almacenamiento de Contraseñas

En lugar de almacenar contraseñas en texto plano, los sistemas almacenan sus hashes. Se utiliza un salt para mejorar la seguridad.

Ejemplo con Salt:

```
C: > Users > beasl > Downloads > ⚙ proceso
1 Contraseña: "password123"
2 Salt: "random123"
3 Contraseña + Salt: "password123random123"
4 Hash SHA-256: 9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08
```

4. Comparativa de Funciones Hash




















Para entender mejor las diferencias entre los principales algoritmos de hash, presentamos una tabla comparativa que incluye características clave como longitud del hash, velocidad, seguridad y usos principales.

Tabla Comparativa de Funciones Hash

CARACTERÍSTICA	MD5	SHA-1	SHA-256
Longitud del Hash	128 bits (32 caracteres)	160 bits (40 caracteres)	256 bits (64 caracteres)
Velocidad	Muy rápida	Rápida	Moderada
Seguridad Actual	Inseguro (vulnerable a colisiones)	Obsoleto (vulnerable a colisiones)	Seguro
Uso Principal	Verificación rápida (no segura)	Firma digital (obsoleto)	Certificados SSL/TLS, blockchain
Ejemplo de Hash	5d41402abc4b2a76b9719d911017c592	aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d	2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c...
Resistencia a Colisiones	Baja	Media	Alta
Adecuado para Criptografía	No	No	Si

Ejemplo Visual:

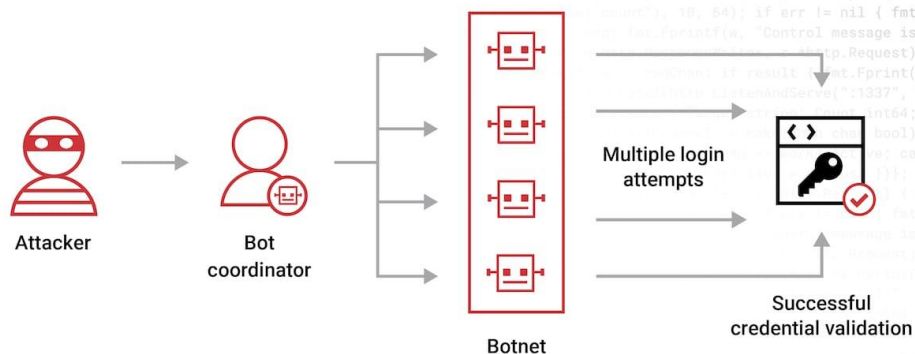
C: > Users > beasl > Downloads > proceso

1	+-----+-----+-----+-----+-----+-----+				
2	Algoritmo	Icono	Longitud Hash	Velocidad	Seguridad
3	+-----+-----+-----+-----+-----+-----+				
4	MD5	 (roto)	[====]		 (Inseguro)
5	SHA-1	 (grieta)	[=====]		 (Obsoleto)
6	SHA-256	 (sólido)	[=====]		 (Seguro)
7	SHA-512	 	[=====]		 (Muy seguro)
8	BLAKE2		[=====]		 (Rápido y seguro)
9	Argon2		[=====]		 (Diseñado para KDF)
10	+-----+-----+-----+-----+-----+-----+				

5. Ataques Avanzados a Funciones Hash

5.1 Ataques de Fuerza Bruta

Los ataques de fuerza bruta intentan generar todas las combinaciones posibles de entradas hasta encontrar una que produzca el mismo hash. Este método es computacionalmente costoso pero efectivo contra funciones hash débiles como MD5.



How a brute-force attack works



5.2 Ataques de Diccionario

En lugar de probar todas las combinaciones posibles, los ataques de diccionario utilizan listas predefinidas de palabras comunes o contraseñas frecuentes para encontrar coincidencias.

1 1 01 0 1 00 011
00 011
1 00 011 0101
1 1 01 0 1 00 011
1 1

Diccionario de
Ciberseguridad

ATAQUES DICCIONARIO

Los ataques de diccionario consisten en **intentar adivinar una contraseña a base a probar las combinaciones** que se han almacenado previamente en un listado, conocido simplemente como diccionario.

5.3 Ataques de Rainbow Tables


Las tablas rainbow son bases de datos precalculadas de hashes para contraseñas comunes. Aunque efectivas, pueden ser mitigadas usando salts .




5.4 Ataques de Colisión

Estos ataques buscan encontrar dos entradas diferentes que produzcan el mismo hash. Ejemplos notables incluyen los ataques a MD5 y SHA-1.


Módulo 7. Funciones hash
Lección 7.4 Colisiones en funciones hash MD5 y SHA-1



Clase c4c7.4
11/05/2021



Colisiones con MD5 en certificados X.509



- "Colliding X.509 Certificates for Different Identities", Lenstra, Wang y Weber (2005) <https://www.win.tue.nl/~bdeweger/CollidingCertificates/>

Class4crypt c4c7.4 - © porgramas 2021 lección 7.4 - página 32

6. Herramientas Modernas de Análisis

6.1 Hashcat

Hashcat es una herramienta potente para romper hashes mediante diferentes tipos de ataques:

- Fuerza bruta : Prueba todas las combinaciones posibles.
- Diccionario : Utiliza listas de contraseñas comunes.
- Máscara : Combina patrones conocidos y valores aleatorios.
- Tablas rainbow : Usa tablas precalculadas de hashes.



6.2 John the Ripper

John the Ripper es otra herramienta popular para auditorías de seguridad. Es especialmente útil para detectar contraseñas débiles en sistemas.



7. Implementaciones en Lenguajes de Programación

7.1 Python

Python incluye la biblioteca **hashlib**, que permite generar hashes fácilmente:

```
C: > Users > beasl > Downloads > ⌵ proceso
1  import hashlib
2
3  texto = "Hola, mundo!"
4  hash_sha256 = hashlib.sha256(texto.encode()).hexdigest()
5  print(hash_sha256)
```

7.2 JavaScript

En JavaScript, puedes usar la API **crypto.subtle** para generar hashes:

```
C: > Users > beasl > Downloads > ⌵ proceso
1  async function generateHash(text) {
2    const encoder = new TextEncoder();
3    const data = encoder.encode(text);
4    const hashBuffer = await crypto.subtle.digest('SHA-256', data);
5    return Array.from(new Uint8Array(hashBuffer))
6      .map(byte => byte.toString(16).padStart(2, '0'))
7      .join('');
8  }
9  generateHash("Hola, mundo!").then(console.log);
```

8. Casos de Uso Innovadores

8.1 Blockchain y Criptomonedas

Las funciones hash son la base de tecnologías como Bitcoin y Ethereum. Cada bloque contiene el hash del bloque anterior, creando una cadena inmutable.



8.2 Identificación de Archivos Únicos

Las funciones hash se utilizan para identificar archivos únicos en sistemas distribuidos como IPFS (InterPlanetary File System).

8.3 Verificación de Datos Médicos

En el sector salud, las funciones hash se utilizan para verificar la integridad de registros médicos electrónicos.

9. Evolución Futura de las Funciones Hash

Con la llegada de la computación cuántica, las funciones hash actuales podrían enfrentar nuevos desafíos. Investigaciones en criptografía post-cuántica están explorando alternativas para garantizar la seguridad en un mundo cada vez más interconectado.

10. Algoritmos de Cifrado: DES y 3DES

10.1 Origen de DES

El algoritmo DES (Data Encryption Standard) fue desarrollado por IBM en la década de 1970 y adoptado como estándar por el gobierno de los Estados Unidos en 1977. Fue diseñado para proteger datos confidenciales mediante un proceso de cifrado simétrico, donde la misma clave se utiliza tanto para cifrar como para descifrar.

10.2 Vulnerabilidades de DES

A pesar de su importancia histórica, DES ha quedado obsoleto debido a sus limitaciones técnicas:

- Clave corta de 56 bits : La longitud de la clave es insuficiente para resistir ataques modernos de fuerza bruta.
- Vulnerabilidad a ataques de hardware avanzado : Con el desarrollo de computadoras más potentes, romper una clave DES se volvió factible en cuestión de horas o días.

Estas debilidades llevaron a la necesidad de una evolución del algoritmo.

10.3 Evolución a 3DES

Para abordar las vulnerabilidades de DES, se desarrolló 3DES (Triple DES) . Este algoritmo aplica el cifrado DES tres veces con claves diferentes, aumentando significativamente la seguridad:

1. Primera etapa : Cifrado con la primera clave.
2. Segunda etapa : Descifrado con una segunda clave.
3. Tercera etapa : Cifrado nuevamente con una tercera clave.

Aunque 3DES proporciona mayor seguridad que DES, sigue siendo menos eficiente en comparación con algoritmos modernos.

10.4 Alternativas Modernas

En respuesta a las limitaciones de DES y 3DES, se han desarrollado nuevos algoritmos de cifrado que ofrecen mayor seguridad y rendimiento:

- AES (Advanced Encryption Standard) : Ampliamente utilizado, admite claves de 128, 192 y 256 bits.
- Blowfish : Un algoritmo rápido y flexible, diseñado para reemplazar DES.
- Twofish : Una versión mejorada de Blowfish, adecuada para aplicaciones de alta seguridad.
- ChaCha20 : Un cifrado de flujo moderno, conocido por su velocidad y eficiencia en dispositivos móviles.

Estos algoritmos son más seguros y eficientes, lo que los convierte en estándares actuales para el cifrado de datos.

10.5 Funcionamiento de DES

El algoritmo DES utiliza un proceso de cifrado en bloques con operaciones específicas para proteger la información. A continuación, se describe su funcionamiento paso a paso:

10.5.1 Bloques de 64 bits

DES divide los datos en bloques fijos de 64 bits para su procesamiento. Si el mensaje no es múltiplo de 64 bits, se añade relleno (padding) para completar el bloque.

10.5.2 16 Rondas

El cifrado se realiza en 16 iteraciones idénticas, cada una utilizando una subclave derivada de la clave principal. Estas rondas combinan operaciones de permutación y sustitución para garantizar la seguridad.

10.5.3 Permutación y Sustitución

DES utiliza dos tipos principales de operaciones:

- Permutación : Reorganiza los bits del bloque según un patrón predefinido.
- Sustitución : Reemplaza grupos de bits con valores diferentes basados en tablas de sustitución.

Estas operaciones aseguran que pequeños cambios en la entrada produzcan grandes cambios en la salida (efecto avalancha).

10.5.4 Cifrado Final

Después de las 16 rondas, se realiza una permutación inversa para obtener el texto cifrado resultante. Este proceso garantiza que el cifrado sea reversible solo con la clave correcta.

Ejemplo Práctico de DES

Supongamos que queremos cifrar el mensaje **"HELLO"** utilizando DES:

1. Entrada : El mensaje se convierte en un bloque de 64 bits.
2. Clave : Se utiliza una clave de 56 bits para generar las subclaves.
3. Procesamiento : El mensaje pasa por 16 rondas de cifrado.
4. Salida : Se obtiene el texto cifrado en formato hexadecimal.

```
C: > Users > beasl > Downloads > ≡ proceso
1  Texto original: "HELLO"
2  Texto cifrado (ejemplo): `3F7C1A2B8D9E0F4A`
```

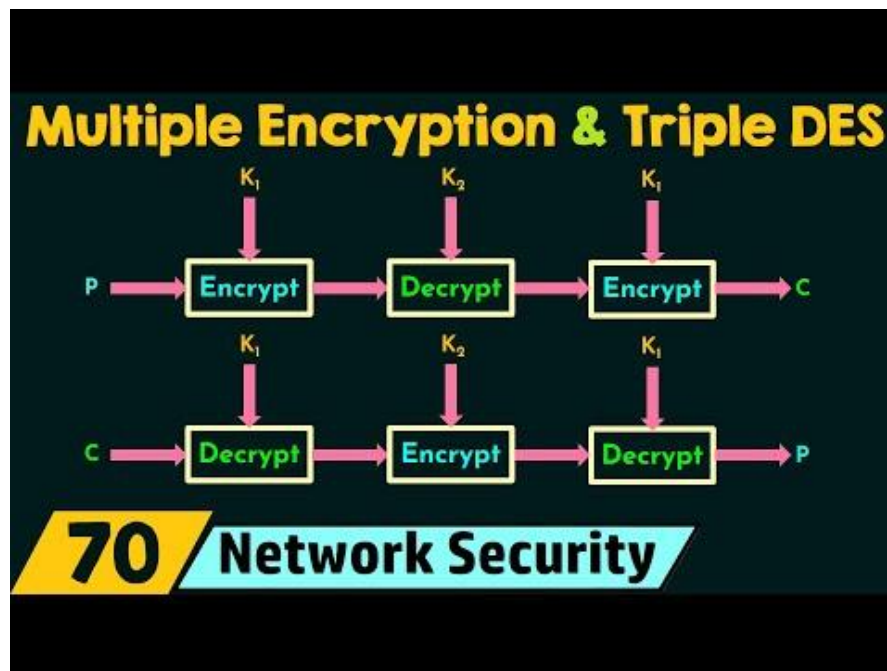
Conclusión

Las funciones hash y los algoritmos de cifrado como DES y 3DES son herramientas fundamentales en la seguridad informática. Aunque DES ha quedado obsoleto debido a sus limitaciones, su evolución a 3DES y la aparición de alternativas modernas como AES han garantizado la protección de datos en entornos digitales. Con el avance de la tecnología, es crucial mantenerse

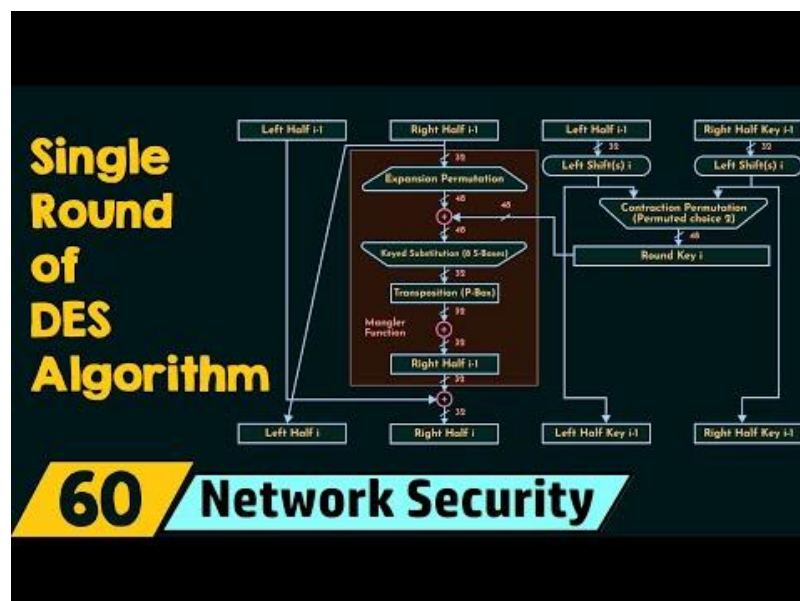
actualizado sobre las mejores prácticas y estándares de cifrado para enfrentar los desafíos de la ciberseguridad.

VIDEOS:

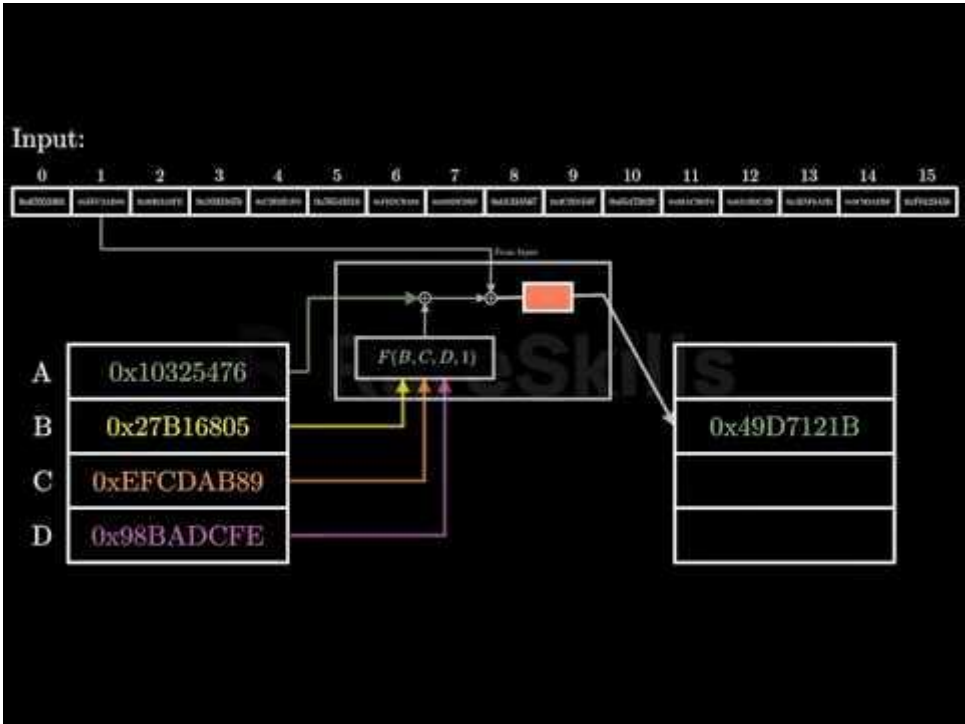
3DES ENCRYPTION



DES Algorithm



MD5 HASH



Webgrafía

1. NIST (National Institute of Standards and Technology)

- Título: "Secure Hash Standard (SHS)"
- URL: <https://csrc.nist.gov/publications/detail/fips/180/4/final>
- Descripción: Documento oficial del NIST que describe los estándares SHA-1, SHA-256 y otros algoritmos hash.

2. OWASP (Open Web Application Security Project)

- Título: "Password Storage Cheat Sheet"
- URL: https://owasp.org/www-project-cheat-sheets/cheatsheets/Password_Storage_Cheat_Sheet.html
- Descripción: Guía sobre buenas prácticas para almacenar contraseñas de forma segura utilizando funciones hash como Argon2 y PBKDF2.

3. MD5 Vulnerabilities

- Título: "The SHattered Attack: Breaking SHA-1 in Practice"
- URL: <https://shattered.io/>
- Descripción: Sitio oficial que documenta el ataque a SHA-1 realizado en 2017, demostrando su vulnerabilidad a colisiones.

4. Cryptographic Hash Functions Explained

- Título: "What is a Cryptographic Hash Function?"
- URL: <https://www.cloudflare.com/learning/ssl/what-is-a-cryptographic-hash/>
- Descripción: Artículo introductorio sobre funciones hash y su uso en seguridad informática.

5. AES Encryption Standard

- Título: "Advanced Encryption Standard (AES)"
- URL: <https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development>

- Descripción: Información oficial del NIST sobre el desarrollo y especificaciones del estándar AES.

6. DES and 3DES Overview

- Título: "Data Encryption Standard (DES) and Triple DES (3DES)"
- URL: <https://www.ibm.com/docs/en/zos/2.4.0?topic=algorithms-data-encryption-standard-des>
- Descripción: Documentación técnica de IBM sobre DES y 3DES, incluyendo su funcionamiento y limitaciones.

7. Hashcat Tool Documentation

- Título: "Hashcat – Advanced Password Recovery"
- URL: <https://hashcat.net/hashcat/>
- Descripción: Página oficial de Hashcat, una herramienta avanzada para pruebas de seguridad relacionadas con hashes.

8. Post-Quantum Cryptography

- Título: "Post-Quantum Cryptography: Preparing for the Future"
- URL: <https://csrc.nist.gov/projects/post-quantum-cryptography>
- Descripción: Proyecto del NIST sobre criptografía post-cuántica y su impacto en algoritmos actuales.

9. Blockchain and Hash Functions

- Título: "How Blockchain Uses Hash Functions"
- URL: <https://www.ibm.com/topics/blockchain/hash-functions>
- Descripción: Artículo sobre el papel de las funciones hash en la tecnología blockchain.

10. Argon2 Password Hashing

- Título: "Argon2: The Winner of the Password Hashing Competition"
- URL: <https://www.argon2.com/>
- Descripción: Sitio oficial de Argon2, un algoritmo moderno diseñado específicamente para almacenamiento seguro de contraseñas.

11. Symmetric Encryption Algorithms

- Título: "Symmetric vs Asymmetric Encryption"
- URL: <https://www.geeksforgeeks.org/difference-between-symmetric-and-asymmetric-key-encryption/>
- Descripción: Comparativa entre cifrados simétricos (como DES y AES) y asimétricos.

12. Rainbow Tables and Attacks

- Título: "Understanding Rainbow Tables"
- URL: <https://www.freecodecamp.org/news/rainbow-tables-explained/>
- Descripción: Explicación detallada sobre qué son las tablas rainbow y cómo se utilizan en ataques a contraseñas.

13. BLAKE2 Hash Function

- Título: "BLAKE2: Fast and Secure Hashing"
- URL: <https://blake2.net/>
- Descripción: Página oficial de BLAKE2, una función hash moderna y rápida diseñada como alternativa a SHA-2 y SHA-3.

14. Python hashlib Documentation

- Título: "hashlib — Secure hashes and message digests"
- URL: <https://docs.python.org/3/library/hashlib.html>
- Descripción: Documentación oficial de Python sobre el módulo **hashlib** para generar hashes.

15. Introduction to Cryptography

- Título: "Cryptography Basics"
- URL: <https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff017e7d:secure-data/xcae6f4a7ff017e7d:cryptography/a/cryptography-basics>
- Descripción: Introducción básica a la criptografía, incluidos conceptos como hashing y cifrado.